Active visual tracking in multi-agent scenarios

by

Yiming Wang

Bachelor of Science (Engineering) 2013

A dissertation submitted to

The School of Electronic Engineering and Computer Science

in partial fulfilment of the requirements for the Degree of

Doctor of Philosophy

in the subject of

Electronic Engineering

Queen Mary University of London

Mile End Road

E1 4NS, London, UK

April 2018

Acknowledgements

I want to primarily thank my supervisor Prof. Andrea Cavallaro for his patience in providing me advice and support during my PhD. I want to thank my second supervisor Dr. Yi-Zhe Song for his continuous encouragement through my PhD especially at the moments of frustration.

I would like to take this opportunity to thank my independent assessor, Dr. Chris Philip for his useful technical suggestions during the internal progress assessments.

I want to thank my parents for their constant moral supports and motivational advice. It is their unconditional love that helped me through all the difficulties.

Finally I want to thank all my friends who worked with me and spent beautiful time with me inside and outside the lab.

This work was jointly supported by Queen Mary University of London and Chinese Scholarship Council.

I, Yiming Wang, confirm that the research included within this thesis is my own work, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Yiming Wang

Date: 15 April 2018

First supervisor | Second supervisor | Author

Prof. Andrea Cavallaro | **Dr. Yi-Zhe Song** | **Yiming Wang**

Active visual tracking in multi-agent scenarios

Abstract

Camera-equipped robots (agents) can autonomously follow people to provide continuous assistance in wide areas, e.g. museums and airports. Each agent serves one person (target) at a time and aims to maintain its target centred on the camera's image plane with a certain size (active visual tracking) without colliding with other agents and targets in its proximity. It is essential that each agent accurately estimates the state of itself and that of nearby targets and agents over time (i.e. tracking) to perform collision-free active visual tracking. Agents can track themselves with either on-board sensors (e.g. cameras or inertial sensors) or external tracking systems (e.g. multi-camera systems). However, on-board sensing alone is not sufficient for tracking nearby targets due to occlusions in crowded scenes, where an external multi-camera system can help. To address scalability of wide-area applications and accurate tracking, this thesis proposes a novel collaborative framework where agents track nearby targets jointly with wireless ceiling-mounted static cameras in a distributed manner. Distributed tracking enables each agent to achieve agreed state estimates of targets via iteratively communicating with neighbouring static cameras. However, such iterative neighbourhood communication may cause poor communication quality (i.e. packet loss/error) due to limited bandwidth, which worsens tracking accuracy. This thesis proposes the formation of coalitions among static cameras prior to distributed tracking based on a marginal information utility that accounts for both the communication quality and the local tracking confidence. Agents move on demand when hearing requests from nearby static cameras. Each agent independently selects its target with limited scene knowledge and computes its robotic control for collision-free active visual tracking. Collision avoidance among robots and targets can be achieved by the Optimal Reciprocal Collision Avoidance (ORCA) method. To further address view maintenance during collision avoidance manoeuvres, this thesis proposes an ORCA-based method with adaptive responsibility sharing and heading-aware robotic control mapping. Experimental results show that the proposed methods achieve higher tracking accuracy and better view maintenance compared with the state-of-the-art methods.

Contents

Published works

[C1] Y. Wang, and A. Cavallaro. Coalition formation with wireless camera networks for distributed tracking. *In proc. of IEEE Int'l Conf. on Advanced Video and Signal-based Surveillance*, Karsruhe, Germany, Aug 2015.

[C2] Y. Wang, and A. Cavallaro. Prioritized target tracking with active collaborative cameras. *In proc. of IEEE Int'l Conf. on Advanced Video and Signal-Based Surveillance*, Colorado Spring, USA, Aug 2016.

[C3] Y. Wang, and A. Cavallaro. Active visual tracking in multi-agent scenarios. *In proc. of IEEE Int'l Conf. on Advanced Video and Signal-Based Surveillance*, Lecce, Italy, Aug 2017.

[C4] Y. Wang, and A. Cavallaro. Concurrent Target following with active directional sensors. *In proc. of IEEE Int'l Conf. on Acoustics, Speech and Signal Processing*, Calgary, Canada, Apr 2018.

[C5] Y. Wang, and A. Cavallaro. Distributed visual tracking for concurrent multi-robot target following. Submitted to *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, Madrid, Spain, Oct 2018.

Electronic preprints are available at *http://www.eecs.qmul.ac.uk/~andrea/publications.html*

Abbreviations and notations

List of Abbreviations

CMOMMT Cooperative Multi-robot Observation of Multiple Moving Targets, page 6

EIF Extended Information Filter, page 17

EIWCF Extended Information-Weighted Consensus Filter, page 19

EKF Extended Kalman Filter, page 17

FoV Field of View of a camera, page 9

ICF Information-weighted Consensus Filter, page 18

ICI Iterative Covariance Intersection, page 19

IF Information Filter, page 16

KCF Kalman Consensus Filter, page 18

KF Kalman Filter, page 15

KLD Tracking-Learning-Detection, page 25

KLT Kanada-Lukas-Tomasi, page 25

ORCA Optimal Reciprocal Collision Avoidance, page 7

PER Packet Error Ratio, page 34

PF Particle Filter, page 18

PID Proportional-Integral-Derivative controller, page 25

PLR Packet Loss Ratio, page 34

PRR Packet Reception Ratio, page 34

ROS Robotic Operating System, page 12

SLAM Simultaneous Localisation And Mapping, page 5

UAV Unmanned Aerial Vehicle, page 25

UKF Unscented Kalman Filter, page 17

VO Velocity Obstacle, page 27

WSCN Wireless Smart Camera Network, page 6

List of symbols

ΔU_i^n Marginal information utility that camera c_i contributes to track o_n, page 36

ε Parameter in consensus fusion, page 19

Λ Set of targets, page 8

ϕ View angle of a camera, page 9

$\mathbf{C}$ Set of cameras, page 8

$\mathbf{F}_n(t)$ Transition matrix of the target state from $t-1$ to t, page 15

$\mathbf{H}_i$(t) Measurement matrix from the target state to the measurement of c_i at t, page 15

$\mathbf{I}_i^n(t)$ Information matrix of the measurement covariance matrix $\mathbf{R}_i^n(t)$, page 16

$\mathbf{i}_i^n(t)$ Information vector of the measurement $\mathbf{z}_i^n(t)$, page 16

$\mathbf{P}_i^n(t)$ Error covariance matrix of the posterior state estimate from c_i for o_n at t, page 15

$\mathbf{P}_i^{n-}(t)$ Error covariance matrix of the prior state estimate from c_i for target o_n at t, page 15

$\mathbf{Q}_n(t)$ Covariance matrix of the process noise for o_n at t, page 10

$\mathbf{R}_i^n(t)$ Covariance matrix of the additive Gaussian noise for measurement of c_i at t, page 10

$\mathbf{X}_i^{n,k}(t)$ Information matrix term that c_i exchanges with neighbours for tracking target o_n at iteration k and t, page 19

$\mathbf{x}_i^{n,k}(t)$ Information vector term that c_i exchanges with neighbours for tracking target o_n at iteration k and t, page 19

$\mathbf{Y}_i^n(t)$ Information matrix of the posterior error covariance matrix $\mathbf{P}_i^n(t)$, page 16

$\mathbf{y}_i^n(t)$ Information vector of the target state estimate $\mathbf{s}_i^n(t)$, page 16

$\mathbf{Y}_i^{n-}(t)$ Information matrix of the prior error covariance matrix $\mathbf{P}_i^{n-}(t)$, page 16

$\mathbf{y}_i^{n-}(t)$ Information state of the prior state estimate $\mathbf{s}_i^{n-}(t)$, page 16

c_i The i^{th} camera, page 8

$f_i(\cdot)$ Transition function of the agent state $\mathbf{s}_i$ from $t-1$ to t, page 9

$f_n(\cdot)$ Transition function of the target state $\mathbf{s}_n$ from $t-1$ to t, page 10

$h_i(\cdot)$ Projection function of c_i from the target state to the measurement, page 10

i Index of a camera, page 8

M Number of cameras, page 8

N Number of targets, page 8

n Index of a target, page 8

o_n The n^{th} target, page 9

r Radius of a robotic platform (target), page 8

t Time step, page 8

$w_n(t)$ Tracking priority of o_n at t, page 9

$\delta_{in}(t)$ Deviation angle from the heading direction of c_i to o_n at t, page 10

$\Lambda_i(t)$ Set of candidate targets that agent c_i receives the request to track at t, page 47

$\mu_i^{n,k}(t)$ Weight for the information from c_i at iteration k using ICI for distributed fusion at t, page 20

$\omega_i(t)$ Rotational speed of the robotic platform for c_i at t, page 9

$\Phi_i(t)$ Set of tracking coalitions that c_i joins in at t, page 33

$\Phi_i^-(t)$ Set of candidate coalitions that c_i joins in at t, page 33

$\mathbf{A}_{i,j}^{*,\tau}(t)$ Set of velocities of agent c_i at t that is closest to its preferred velocity and reciprocally collision-avoiding to c_j in τ time steps, page 62

$\mathbf{A}_i^{*,\tau}(t)$ Set of accessible velocities of agent c_i at t that are collision-avoiding to nearby agents and targets in τ time steps, page 63

$\mathbf{C}_n^{m,R}(t)$ Set of agents that receive the request to track target o_n at t, page 47

$\mathbf{C}^m$ Set of camera-equipped agents, page 8

$\mathbf{C}^s$ Set of static cameras, page 8

$\mathbf{C}_i(t)$ Neighbouring cameras of c_i at t, page 8

$\mathbf{C}_i^n(t)$ Set of neighbouring cameras of c_i that are in the coalition for tracking o_n at t, page 36

$\mathbf{C}_n(t)$ Set of cameras that are in the coalition for tracking target o_n at t, page 32

$\mathbf{p}_i(t)$ Position of c_i at t, page 9

$\mathbf{p}_n(t)$ Position of o_n at t, page 9

$\mathbf{s}_i^{n-}(t)$ Predicted (prior) state estimate of o_n from camera c_i at t, page 15

$\mathbf{s}_i(t)$ State of c_i at t, page 9

$\mathbf{s}_n(t)$ State of o_n at t, page 9

$\mathbf{u}_i(t)$ Robotic control vector of agent c_i at t, page 9

$\mathbf{v}_i^*(t)$ Preferred velocity of agent c_i at t, page 60

$\mathbf{v}_i(t)$ Velocity of c_i at t, page 9

$\mathbf{v}_i^A(t)$ Collision-avoiding velocity of agent c_i at t, page 63

$\mathbf{v}_n(t)$ Velocity of o_n at t, page 9

$\mathbf{z}_i^n(t)$ Measurement of c_i corresponding to the target state $\mathbf{s}_n(t)$, page 9

$\theta_i(t)$ Camera orientation (heading of the robotic platform) of c_i at t, page 9

$\varsigma_i^n(t)$ Additive Gaussian noise of measurement of c_i at t, page 10

$\xi_i(t)$ Process noise of agent c_i at t, page 9

$\xi_n(t)$ Process noise of target o_n at t, page 10

$a_{i,j}(t)$ Responsibility that agent c_i shares with c_j at t, page 62

$d_{ij}(t)$ Distance between c_i and c_j at t, page 8

$d_{in}(t)$ Distance between c_i and o_n at t, page 10

d_{in}^* Desired distance between an agent c_i and its target o_n, page 10

r_c Communication range of a camera, page 8

r_v View range of a camera, page 9

$U_i^n(t)$ Information utility that camera c_i contributes to tracking target o_n at t, page 34

$v_i(t)$ Translational speed control of the robotic platform for c_i at t, page 9

Chapter 1

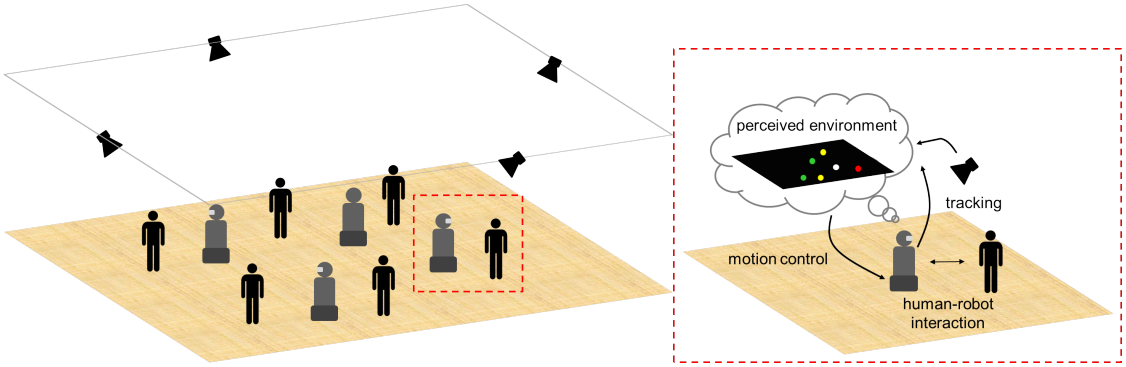
Introduction

1.1 Motivation

Service robots that are designed for assisting people have been developed by both academics and industries [22]. For example, tour-guide robots, Minerva [131] and RoboX [56], have been developed to serve in museums or at expos to engage and guide people through exhibits. Instead of being guided by a robot, a person may want to navigate in a public place as he/she wishes while being followed by a robot that provides continuous assistance. For example, a person can walk in a museum freely and request to the following robot for the history of a painting whenever the person comes across an interesting one. Prototypes of person-following robots, such as ApriAttenda [142] and Kompai [48] have been developed to serve at home or in public places. Commercial robots, such as Pepper from Softbank with ongoing and open development, have been recently deployed in shopping malls for entertaining and serving people[1].

In general, the existing prototypes and commercial robots serve people either within a rather constrained area, e.g. a Pepper robot interacts with people in shopping malls without moving around, or without coexisting with other robots that are serving in the same environment [131, 142, 46, 52, 48, 32]. This thesis considers a camera-equipped robot as an agent and focuses on developing an autonomous multi-agent system in public places, where each agent follows one person (target) to provide continuous assistance and multiple agents concurrently serve their own target in the same environment without colliding with nearby agents and targets (the avoidance

[1] https://www.youtube.com/watch?v=FhWAsnPOn_w Last accessed: 30/08/2017



(a) (b)

Figure 1.1: Multiple agents coexist in a shared environment with each agent actively tracking and serving a target person. External ceiling-mounted static cameras assist agents for tracking nearby targets. (a) An example scenario with four agents serving in a crowd with four ceiling-mounted cameras. (b) Illustration of the three main tasks for one agent: tracking itself and nearby targets/agents, motion control using the perceived environment and human-robot interaction.

of static obstacles is not considered in this thesis) (see Fig. 1.1(a)). Three main tasks for such multi-agent automation are identified: i) tracking agents and targets, ii) motion control with the perceived environment and iii) human-robot interactions (see Fig. 1.1(b)).

By tracking we mean estimating the states of agents and targets in a reference 2D coordinate over time. The state of an agent consists of pose (position and orientation) and velocity. The state of a target consists of position and velocity. Motion control refers to an agent computing its robotic control vector at each time in order to accomplish its assigned task [71, 69]. In our case, the task of an agent is following its target without colliding with nearby agents and targets. Finally, human-robot interaction means that an agent perceives the intention or emotion of its target person with sensory inputs and establishes friendly interactions [122]. This thesis focuses on tracking and motion control parts, while human-robot interaction is beyond the scope.

Each agent can track itself, nearby agents and targets with either on-board sensors or external sensors, such as ceiling-mounted camera(s) [57, 127, 104, 11]. For agent self-tracking, wheel/visual odometry techniques can be used with on-board sensors, such as wheel encoders, an Inertial Measurement Unit (IMU) and a camera [27, 118, 39, 28]. Odometry is generally prone to drifts and inaccurate measurements when traversing uneven terrain. Therefore other on-board sensors such as LiDAR, and other techniques such as Simultaneous Localisation And Mapping (SLAM), can be used to improve the agent self-tracking accuracy [19, 9, 83, 31, 121, 144]. This thesis assumes that agents are able to track themselves and exchange their states with neighbouring agents [51].

Due to occlusions in scenes populated by targets and agents, it is not trivial for an agent to achieve accurate and consistent tracking of nearby targets only using on-board sensing. External cameras can help to address the occlusion problem and improve the tracking accuracy by exploiting different view perspectives of the scene [21, 104]. For the intended applications that require tracking in wide areas, a wireless smart camera network (WSCN) can be employed. Each smart camera in WSCN is capable of real-time on-board vision sensing/processing and communicating with other smart cameras [15, 111].

There can be three network designs to achieve tracking with a WSCN; namely, centralised, decentralised and distributed. With centralised tracking, all smart cameras send measurements to a central node that is responsible for tracking all targets [21]. With decentralised tracking, each target is tracked by a subset of smart cameras at a time. One smart camera within the subset works as a local centre to coordinate other smart cameras and to perform tracking with the measurements provided by the smart cameras within the subset [106, 33, 117, 40, 129, 13]. With distributed tracking, each smart camera iteratively exchanges and updates its local state estimates with one-hop neighbouring smart cameras in order to achieve agreed state estimates of targets [58, 59, 61].

Centralised design is not scalable as centralised processing is bandwidth demanding and computationally expensive for the central node and can lead to system failures if the central node fails. Decentralised and distributed tracking frameworks are preferable with WSCNs for scalability [1]. With decentralised tracking, only local centres have target state estimates at each time, while distributed tracking enables each smart camera with target state estimates via iterative neighbourhood communication. Typical distributed tracking methods involve all cameras in a network [58, 59, 61]. When tracking many targets with a bandwidth-limited WSCN, such as a network with IEEE 802.15.4 standard [103], involving all cameras can deteriorate tracking accuracy due to poor communication quality, i.e. with packet loss and packet errors. Solutions to mitigate this problem is to form coalitions among static smart cameras prior to distributed tracking, thus keeping the information localised and increasing information delivery reliability [C1].

Agents can perform distributed tracking for nearby targets jointly with their neighbouring static smart cameras. With the states of nearby targets and agents, agents independently select their target to follow. One closely related problem is the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) problem whose main objective is to maximise the on-

target observation time [100, 99, 35, 68, 63, 10]. CMOMMT focuses on scenarios where there are more targets than agents and one agent is preferred to cover multiple targets, which is different from our intended applications. CMOMMT treats each target with the same importance, i.e. each target has the same priority. In this thesis, we assume that each target is followed with a different priority. For example, a person with movements that show anxiety should have a higher priority to be served compared to a person moving normally. This thesis designs a neighbourhood messaging scheme between static smart cameras and agents, and uses a local target selection criterion to achieve a one-to-one target-agent assignment. The objective of the assignment is to maximise the prioritised on-target observation time, i.e. the on-target observation time weighted by the priority [C2].

Each agent follows its target with the primary objective of maintaining the target centred on the image plane with a certain size, which is termed as active visual tracking in this thesis. When there are multiple agents performing active visual tracking concurrently, agents need to avoid collisions among themselves and with targets. Agents can be considered as moving obstacles that are reactive to the scene dynamics. Reciprocity is important for multi-agent collision avoidance to avoid undesirable oscillations [12]. One of the most used strategies to achieve reciprocity is the Optimal Reciprocal Collision Avoidance (ORCA) method [12]. ORCA has been successfully validated with both simulations and robotic platforms, such as iCreate [11] and AR drones [4], during the past two decades. ORCA assumes that each agent applies the same collision avoidance strategy, is able to sense the exact shapes, positions and velocities of nearby moving obstacles and infer their preferred velocity. The preferred velocity is the one that a moving obstacle would move at as if there were no entities in its way. Each agent firstly identifies a set of velocities that will not lead to collisions with any nearby moving obstacles, and then selects a new collision-free velocity from the set that is closest to its preferred velocity. Most works apply ORCA for multi-robot navigation or crowd simulations where each agent, i.e. a robot or a simulated person, navigates to a given goal position [12, 127, 128, 5, 11]. The paths of navigating agents are rather flexible during collision avoidance as long as agents reach their goal position. When applying ORCA to agents that perform active visual tracking, collision avoidance should further account for maintaining the on-target view performance, which to the best of our knowledge has not been explored in the literature [C4].

The following sections formalise the problem and present the detailed technical contributions.

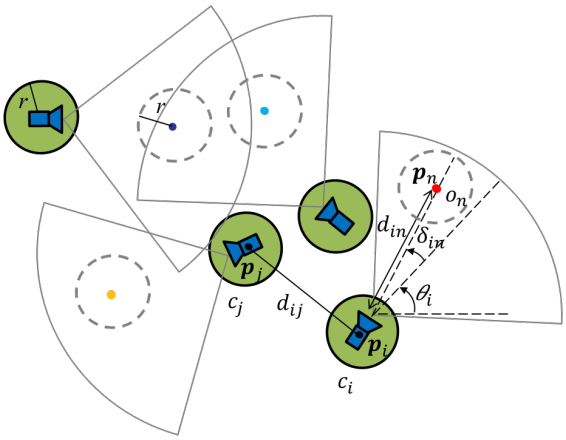


Figure 1.2: Top-view of multiple disk-shaped agents actively tracking their point-like target. Agent c_i of radius r locates at $\mathbf{p}_i$ with heading direction of θ_i and point-like target o_n locates at $\mathbf{p}_n$. Targets are differentiated by colours and each target is considered with a disk shape of the same radius as the agent during collision avoidance (indicated by grey circles in dashed lines). d_{ij} is the distance between agent c_i and another agent c_j. d_{in} is the distance from c_i to its target o_n and δ_{in} is the deviation angle from the heading of the agent to its target.

1.2 Problem formulation

Let a planar area be covered by a set of M cameras $\mathbf{C} = \{c_1, c_2, ..., c_i, ..., c_M\}$. Each camera is a smart camera and the area is free of static obstacles. Let there be a set of N targets $\Lambda = \{o_1, o_2, ..., o_n, ..., o_N\}$ moving in this area. This thesis uses i to index the cameras, n to index the targets and t to index the time steps.

$\mathbf{C}$ can be presented as $\mathbf{C} = \mathbf{C}^s \cup \mathbf{C}^m$, where $\mathbf{C}^s$ is the set of static cameras and $\mathbf{C}^m$ is the set of camera-equipped agents. Static cameras are fixed at a ceiling level with a top-down view on the area. Agents move within the planar area. All cameras (including the static cameras and the cameras equipped on agents) are synchronised, calibrated and connected with other cameras wirelessly. This thesis omits the superscript m or s when referring to generic cameras.

Let r_c be the communication range of a camera. The neighbours of c_i at t, $\mathbf{C}_i(t)$, is the set of cameras within the communication range of c_i:

$$\mathbf{C}_i(t) = \{c_j,\ d_{ij}(t) < r_c\}, \tag{1.1}$$

where $d_{ij}(t)$ is the distance between c_i and c_j at t. Each c_i only communicates with its neighbours $\mathbf{C}_i(t)$.

Each agent c_i is composed of a disk-shaped robotic platform with radius r and a camera with

a sector-shaped field of view (FoV) defined by the viewing angle ϕ and viewing range r_v. Let $\theta_i(t)$ be the camera orientation of c_i at t which is same as the heading of the robotic platform (see Fig. 1.2). Let $\mathbf{s}_i(t)$ be the state of agent c_i in a 2D global coordinate system at t:

$$\mathbf{s}_i(t) = \left[x_i(t),\ y_i(t),\ \theta_i(t),\ \dot{x}_i(t),\ \dot{y}_i(t),\ \dot{\theta}_i(t)\right]^T, \tag{1.2}$$

where $\mathbf{p}_i(t) = [x_i(t),\ y_i(t)]^T$ is the position and $\mathbf{v}_i(t) = [\dot{x}_i(t),\ \dot{y}_i(t)]^T$ is the velocity of c_i. The superscript T indicates the transpose of a matrix or vector.

The state of an agent is time variant. Let the state transition of an agent be:

$$\mathbf{s}_i(t) = f_i(\mathbf{s}_i(t-1), \mathbf{u}_i(t)) + \xi_i(t), \tag{1.3}$$

where $f_i(\cdot)$ is the transition function of agent state $\mathbf{s}_i$ from $t-1$ to t and $\xi_i(t)$ is the process noise. $f_i(\cdot)$ follows non-linear kinematic models, e.g. car-like or differential-drive models [128, 5]. A car-like model updates the state of a robotic platform by changing the steering angle and speed of the front wheels while a differential-drive model updates the state by changing independently the speed of wheels. $\mathbf{u}_i(t)$ is the robotic control vector of agent c_i and consists of a translational speed $v_i(t)$ and a rotational speed $\omega_i(t)$. $\mathbf{u}_i(t)$ is bounded as $|v_i(t)| \leqslant v_{\max}$ and $|\omega_i(t)| \leqslant \omega_{\max}$, where $v_{\max}$ and $\omega_{\max}$ are the maximum translational speed and rotational speed, respectively. Each agent is aware of its own state and that of its neighbouring agents.

Targets are identifiable by discriminative features (e.g. colour or fiducial markers) for each camera. This thesis models each target with a disk shape of the same radius as that of the robotic platform for collision avoidance. Each target o_n is associated with a tracking priority $w_n(t) \in [0,1]$ that is provided by static cameras. Priority equals to one when the target is required to be tracked for the longest time while priority equals to zero when there is no need for tracking a target. The priority is potentially time-varying, but considered as constant in this thesis.

Let $\mathbf{s}_n(t)$ be the state of o_n in the 2D global coordinate system:

$$\mathbf{s}_n(t) = \left[x_n(t),\ y_n(t),\ \dot{x}_n(t),\ \dot{y}_n(t)\right]^T, \tag{1.4}$$

where $\mathbf{p}_n(t) = [x_n(t),\ y_n(t)]^T$ is the target position and $\mathbf{v}_n(t) = [\dot{x}_n(t),\ \dot{y}_n(t)]^T$ is the velocity of target. Let $\mathbf{z}_i^n(t)$ be the measurement of c_i corresponding to the target state $\mathbf{s}_n(t)$ and modelled

as:

$$\mathbf{z}_i^n(t) = h_i(\mathbf{s}_n(t)) + \varsigma_i^n(t), \tag{1.5}$$

where $\varsigma_i^n(t)$ is an additive Gaussian noise with zero mean and covariance $\mathbf{R}_i^n(t)$. $h_i(\cdot)$ is the projection function from the target state to the measurement. We assume that there is no false positives (measurements that do not belong to targets) but false negatives (missing measurements that belong to targets) may occur.

Let the target dynamics in one time step be:

$$\mathbf{s}_n(t) = f_n(\mathbf{s}_n(t-1), \xi_n(t)), \tag{1.6}$$

where $f_n(\cdot)$ is the transition function of the target state from $t-1$ to t and $\xi_n(t)$ is the process noise which follows a Gaussian distribution with zero mean and covariance matrix $\mathbf{Q}_n(t)$.

One target is tracked by at most one agent and one agent tracks only one target at a time. By communicating with nearby static cameras, each agent c_i locally selects a target to track with the objective of maximising the prioritised observation time.

Let $d_{in}(t)$ be the distance between c_i and o_n at t, and $\delta_{in}(t)$ be the deviation angle from the heading direction of c_i to o_n at t (see Fig. 1.2). Agent c_i computes the robotic control vector $\mathbf{u}_i(t)$ to maintain o_n at a certain distance, d_{in}^*, in the heading direction of the robotic platform, i.e. $d_{in}(t) = d_{in}^*$ and $\delta_{in}(t) = 0$, while avoiding collisions with any other agents or any targets at any time, i.e. $d_{ij}(t) > 2r, \forall j,\ j \neq i$ and $d_{in}(t) > 2r, \forall n$ at any t.

1.3 Contributions

This thesis proposes a collaborative framework for multi-agent active visual tracking in wide areas with the assistance of a WSCN for target tracking. At each time step, static cameras dynamically form tracking coalitions with the consideration of communication imperfections. Distributed tracking is performed within each tracking coalition together with agents in the proximity, so that agents can obtain the fused state estimates of nearby targets. On hearing requests from static cameras, agents independently select a target to follow with the objective of maximising the prioritised on-target observation time. Agents compute their control vector to actively track their selected target with view maintenance while avoiding collisions with nearby agents and tar-

gets. The main technical contributions with respect to the state-of-the-art methods are described as follows.

Communication-aware coalition formation for distributed tracking

Distributed tracking allows each agent to achieve agreed state estimates of its nearby targets via iteratively communicating with neighbouring static cameras. Typical distributed tracking methods involve all cameras in a network, which can deteriorate tracking accuracy when there are many targets to track and the network bandwidth is limited [58, 59, 61]. This thesis proposes a communication-aware coalition formation scheme that consists of a three-stage neighbourhood messaging. The scheme enables each static camera to independently decide its tracking coalitions based on the marginal utility that accounts for both the local tracking confidence and the communication link quality under realistic modelling [C1].

Motion control with view maintenance and energy efficiency

Each agent selects one target to perform active visual tracking with the objective of maximising the prioritised on-target observation time. Such assignment of agents to targets is related to the CMOMMOT problem but different as the agent in the CMOMMT problem is assigned to multiple targets without accounting for the priority of target [100, 99, 35, 68, 63, 10]. This thesis proposes a target selection scheme where agents compute a local utility for each of their candidate targets and select the target with the highest utility. The local utility accounts for the target tracking priority, a distance-based cost and a factor that aims to reduce the chances of multiple agents selecting the same target. Once a target is associated, the agent computes its robotic control by minimising a weighted cost functions that accounts for both view maintenance and energy efficiency. The trade-off between the two costs is investigated for optimal weight tuning [C2].

Collision avoidance with view maintenance

Collision avoidance among agents and targets can be achieved by the ORCA method, which has been mostly applied for multi-agent navigation or crowd simulations [12, 127, 128, 5, 11]. When applying ORCA to agents that perform active visual tracking, the collision-free paths are further constrained by the dynamics of targets and the limited FoV of an on-board camera. This thesis proposes an ORCA-based method that addresses view maintenance during collision avoidance manoeuvres. Agents that are in danger of losing their target out of the FoV should share less responsibility for collision avoidance so that they can

move at their preferred velocity for view maintenance. An adaptive responsibility sharing algorithm is proposed to set the pair-wise responsibility based on the difference between the preferred velocity and the current velocity of an agent. In order to maintain the agent heading towards its target, the proposed heading-ware robotic control mapping algorithm further takes into account the deviation angle from the agent's heading to its target when computing the robotic control from the collision-free velocity [C4].

Robotic validation of multi-agent active visual following

This thesis implements a scalable and fully distributed multi-agent system for concurrent active visual following without any central processing. Each agent estimates the states of all agents and targets via distributed tracking with static cameras, and derives its own collision-free robotic control on board. We design and implement the system on Robotic Operating System (ROS) with off-the-shelf web cameras and robotic platforms. The proposed implementation addresses practical challenges such as wireless communication imperfections, including packet delay and loss, caused by interferences and concurrent transmission, and tracking failures caused by detection or transmission errors. The robotic implementation runs in real time and is validated in terms of view maintenance, collision avoidance and system latency [C5].

1.4 Organisation of thesis

This thesis is organised as follows.

Chapter 1: This Chapter introduces the problem of multi-agent active visual tracking in wide areas. The problem is formulated and contributions are listed.

Chapter 2: This Chapter presents the state of the art on distributed tracking with WSCNs in Section 2.1 and motion control among multiple agents for active visual tracking in Section 2.2. Section 2.3 summarises this Chapter.

Chapter 3: This Chapter presents the proposed collaborative framework for multi-agent active visual tracking. The proposed framework is introduced in Section 3.2. The method evaluation with simulations is presented in Section 3.3. Section 3.4 finally summarises this Chapter.

Chapter 4: This Chapter presents the local decision-making and motion control for multi-agent active visual tracking. Section 4.2 describes the proposed method; Section 4.3 presents the method evaluation with simulations; Finally, Section 4.4 summarises this chapter.

Chapter 5: This Chapter presents the collision avoidance among multiple agents performing active visual tracking. Section 5.2 introduces the proposed method. Section 5.3 validates the proposed method with simulations using real people trajectories extracted from public datasets. Section 5.4 finally concludes this Chapter.

Chapter 6: This Chapter presents the implementation of the distributed multi-agent tracking and control system. Section 6.2 describes the proposed infrastructure on ROS and the implementation of each functionality. Section 6.3 presents the experimental setup and results. Finally, conclusion is drawn in Section 6.4.

Chapter 7: This Chapter first summarises the achievements in Section 7.1 and presents future research directions in Section 7.2.

Chapter 2

State of the art

This Chapter reviews the state-of-the-art works related to distributed tracking with WSCNs and multi-agent active visual tracking. For the literature of tracking with a WSCN, the focus is on distributed tracking techniques and camera coalitions for tracking with resource constraints. While works that address other vision-related challenges, such as occlusions and measurements association in multi-object tracking, are beyond the scope but are referred to [135, 80, 112]. For multi-agent active visual tracking, three related areas are covered: i) the assignment of multiple agents to multiple targets, ii) motion control for active visual tracking with a single agent and iii) collision avoidance among multiple agents.

2.1 Distributed tracking with wireless smart camera networks

Distributed tracking enables each camera with an agreed state estimate of a target via exchanging information with one-hop neighbours. Each camera performs two operations which are local filtering and distributed fusion at each time step. With measurements, each camera first locally updates the state estimate of a target using filtering techniques and then prepares information to exchange with neighbours using distributed fusion techniques. When the network scale or the number of targets increases, distributed tracking can be demanding in terms of resources, e.g. energy and bandwidth, as distributed fusion involves iterative neighbourhood communication. It is essential to improve the resource efficiency by forming dynamic groups of cameras, i.e. coalitions, for tracking targets. The following sections discuss in details the works on filtering techniques, distributed fusion techniques and camera coalition formation for tracking targets.

2.1.1 Filters for state estimation

Bayesian filter is a probabilistic approach that runs at each camera for sequentially estimating the unknown probability density function of the target state using the current measurements and a process model. The true state is assumed to be an unobserved Markov process, and the measurements are the observations of a hidden Markov model. Kalman Filter (KF) is a recursive Bayesian filter for linear dynamic system with the yielded conditional probabilities and all noises following Gaussian distribution. KF has been widely applied in both object tracking [95, 58, 62] and robot navigation [81, 121, 9, 25]. KF estimates the target state corresponding to the current time step by first predicting an estimate with the previous target state estimate and then updating/correcting the predicted target state estimate with the current measurement.

Let the predicted (prior) state estimate of o_n from c_i at t be $\mathbf{s}_i^{n-}(t)$. The prior state estimate is computed from the state transition function as in Eq. 1.6. In the case of a linear transition function, there exists a transition matrix of target state, $\mathbf{F}_n(t)$, such that $f_n(\mathbf{s}_i^n(t-1)) = \mathbf{F}_n(t)\mathbf{s}_i^n(t-1)$. KF computes $\mathbf{s}_i^{n-}(t)$ and its corresponding error covariance matrix $\mathbf{P}_i^{n-}(t)$ as:

$$\begin{aligned} \mathbf{s}_i^{n-}(t) &= \mathbf{F}_n(t)\mathbf{s}_i^n(t-1) \\ \mathbf{P}_i^{n-}(t) &= \mathbf{F}_n(t)\mathbf{P}_i^n(t-1)\mathbf{F}_n(t)^T + \mathbf{Q}_n(t), \end{aligned} \tag{2.1}$$

where $\mathbf{P}_i^n(t-1)$ is the error covariance matrix of the target state estimate at $t-1$.

KF then updates the target state estimate with the current measurement $\mathbf{z}_i^n(t)$. When the measurement model in Eq. 1.5 is linear, there will be a measurement matrix, $\mathbf{H}_i(t)$, such that $h_i(\mathbf{s}_i^n(t)) = \mathbf{H}_i(t)\mathbf{s}_i^n(t)$. The state estimate $\mathbf{s}_i^n(t)$ and its corresponding error covariance matrix $\mathbf{P}_i^n(t)$ are updated as:

$$\begin{aligned} \mathbf{s}_i^n(t) &= \mathbf{s}_i^{n-}(t) + \mathbf{K}_i^n(t)\left[\mathbf{z}_i^n(t) - \mathbf{H}_i(t)\mathbf{s}_i^{n-}\right] \\ \mathbf{P}_i^n(t) &= \left[\mathbf{I} - \mathbf{K}_i^n(t)\mathbf{H}_i(t)\right]\mathbf{P}_i^{n-}(t), \end{aligned} \tag{2.2}$$

where $\mathbf{I}$ is the identity matrix of the same dimension of $\mathbf{P}_i^n(t)$ and $\mathbf{K}_i^n(t)$ is known as the Kalman gain which tells how much contribution the correction from the measurement is made to the prior state estimate. The optimal Kalman gain is computed as [138]:

$$\mathbf{K}_i^n(t) = \mathbf{P}_i^{n-}(t)\mathbf{H}_i(t)^T\left[\mathbf{H}_i(t)\mathbf{P}_i^{n-}(t)\mathbf{H}_i(t)^T + \mathbf{R}_i^n(t)\right]^{-1}. \tag{2.3}$$

An alternative form of KF is Information Filter (IF) that uses the inverse covariance for estimation. IF is mathematically equivalent to KF (proof is referred to [64]) with the computational complexity increased in the prediction and reduced in the update, compared to KF. This makes IF commonly used in sensor fusion as the update using measurement information from multiple sources can be achieved by trivial additions [62].

IF represents the target state estimate and its corresponding covariance matrix as the information vector $\mathbf{y}_i^n(t)$ and information matrix $\mathbf{Y}_i^n(t)$:

$$\begin{aligned} \mathbf{Y}_i^n(t) &= \mathbf{P}_i^n(t)^{-1} \\ \mathbf{y}_i^n(t) &= \mathbf{P}_i^n(t)^{-1}\mathbf{s}_i^n(t). \end{aligned} \tag{2.4}$$

The information vector $\mathbf{y}_i^{n-}(t)$ and matrix $\mathbf{Y}_i^{n-}(t)$ corresponding to the prior state estimate can be computed in the information form as:

$$\begin{aligned} \mathbf{y}_i^{n-}(t) &= \mathbf{Y}_i^{n-}(t)\mathbf{F}_n(t)\mathbf{Y}_i^n(t-1)^{-1}\mathbf{y}_i^n(t-1) \\ \mathbf{Y}_i^{n-}(t) &= \mathbf{M}_i^n(t) - \mathbf{M}_i^n(t)\left[\mathbf{Q}_n(t)^{-1} + \mathbf{M}_i^n(t)\right]^{-1}\mathbf{M}_i^n(t), \end{aligned} \tag{2.5}$$

where $\mathbf{M}_i^n(t) = \mathbf{F}_n(t)^{-T}\mathbf{Y}_i^n(t-1)\mathbf{F}_n(t)^{-1}$. Alternatively, $\mathbf{y}_i^{n-}(t)$ and $\mathbf{Y}_i^{n-}(t)$ can also be computed from KF as:

$$\begin{aligned} \mathbf{Y}_i^{n-}(t) &= \mathbf{P}_i^{n-}(t)^{-1} \\ \mathbf{y}_i^{n-}(t) &= \mathbf{P}_i^{n-}(t)^{-1}\mathbf{s}_i^{n-}(t). \end{aligned} \tag{2.6}$$

IF computes the measurement information vector $\mathbf{i}_i^n(t)$ and information matrix $\mathbf{I}_i^n(t)$:

$$\begin{aligned} \mathbf{i}_i^n(t) &= \mathbf{H}_i(t)^T\mathbf{R}_i^n(t)^{-1}\mathbf{z}_i^n(t) \\ \mathbf{I}_i^n(t) &= \mathbf{H}_i(t)^T\mathbf{R}_i^n(t)^{-1}\mathbf{H}_i(t). \end{aligned} \tag{2.7}$$

$\mathbf{I}_i^n(t)$ and $\mathbf{i}_i^n(t)$ are set to zero matrix $\mathbf{0}$ of the same dimension of $\mathbf{P}_i^n(t)$ and $\mathbf{s}_i^n(t)$, respectively, when c_i does not have any measurement of o_n at t.

With both information from the measurement and prior target state estimate, the information state $\mathbf{y}_i^n(t)$ and its corresponding information matrix $\mathbf{Y}_i^n(t)$ corresponding to the target state

estimate is finally obtained by addition:

$$\begin{aligned}\mathbf{y}_i^n(t) &= \mathbf{y}_i^{n-}(t) + \mathbf{i}_i^n(t) \\ \mathbf{Y}_i^n(t) &= \mathbf{Y}_i^{n-}(t) + \mathbf{I}_i^n(t).\end{aligned} \tag{2.8}$$

KF and IF are used for linear systems. In the case of visual tracking, the projection of a 2D image-plane measurement to a target state in the world coordinate is non-linear and the process models of people and robotic platforms are mostly non-linear as well. When the dynamic system is non-linear, i.e. the state transition function ($f_n(\cdot)$) or the measurement transition function ($h_i(\cdot)$) is non-linear, linear filters cannot be used [61]. To address the non-linearity, several variants of KF have been proposed, such as the Extended Kalman Filter (EKF) [124] and Unscented Kalman Filter (UKF) [125].

EKF linearises at the current state estimate by using partial derivatives, i.e. the Jacobian of the non-linear functions $f_n(\cdot)$ and $h_i(\cdot)$ in the KF equations. Extended Information Filter (EIF) is the corresponding information form of EKF, which is the most commonly applied filter in sensor fusion with non-linear dynamic systems. The prediction step for EIF is similar to the equations of IF in Eq. 2.5 as:

$$\begin{aligned}\mathbf{y}_i^{n-}(t) &= \mathbf{Y}_i^{n-}(t) f(\mathbf{Y}_i^n(t-1)^{-1} \mathbf{y}_i^n(t-1)) \\ \mathbf{Y}_i^{n-}(t) &= \mathbf{M}_i^n(t) - \mathbf{M}_i^n(t) \left[\mathbf{Q}_n(t)^{-1} + \mathbf{M}_i^n(t)\right]^{-1} \mathbf{M}_i^n(t),\end{aligned} \tag{2.9}$$

where $\mathbf{M}_i^n(t) = \mathbf{J}_{f,n}(t)^{-T} \mathbf{Y}_i^n(t-1) \mathbf{J}_{f,n}(t)^{-1}$ and $\mathbf{J}_{f,n}(t)$ is the Jacobian of $f_n(\cdot)$ with respect to $\mathbf{s}_i^n(t)$.

EIF computes $\mathbf{i}_i^n(t)$ and $\mathbf{I}_i^n(t)$ as:

$$\begin{aligned}\mathbf{i}_i^n(t) &= \mathbf{J}_{h,i}(t)^T \mathbf{R}_i^n(t)^{-1} \left[\mathbf{z}_i^n(t) - h_i(\mathbf{s}_i^{n-}(t)) + \mathbf{J}_{h,i}(t) \mathbf{s}_i^{n-}(t)\right] \\ \mathbf{I}_i^n(t) &= \mathbf{J}_{h,i}(t)^T \mathbf{R}_i^n(t)^{-1} \mathbf{J}_{h,i}(t),\end{aligned} \tag{2.10}$$

where $\mathbf{J}_{h,i}(t)$ is the Jacobian of $h_i(\cdot)$ with respect to $\mathbf{s}_i^n(t)$. The update is the same as in Eq. 2.8.

EKF can achieve accurate estimation if the functions are close to linear functions and the uncertainties of measurements and prior estimates are small [124]. As for highly non-linear systems, UKF can be applied by picking a minimal set of sample points around the mean which

are then propagated via the non-linear functions to form a new mean and covariance. UKF avoids the calculation of the Jacobian which can be difficult to obtain for highly non-linear functions. Other filters, such as Particle Filter (PF) [8], also known as Sequential Monte Carlo method, also address non-linearity by using a set of particles to represent the distribution instead of using a Gaussian distribution assumption.

2.1.2 Distributed fusion of local estimates

Distributed fusion techniques are traditionally used in wireless sensor networks to achieve a global agreement on an environmental phenomenon, e.g. temperature, via iteratively exchanging information with neighbours [95]. Distributed fusion algorithms define how the information is exchanged and locally processed. Basic schemes for disseminating information within a network include flooding and gossiping [2, 133]. Flooding allows each sensor broadcast its local knowledge to neighbours blindly whenever there is an update that either comes from itself or is received from its neighbours. Flooding can be resource-consuming in large-scale networks due to the duplicated information transmissions [113]. Gossiping is developed upon flooding but allows each sensor to randomly select one neighbour to exchange information at each time [14, 107].

For distributed tracking, one of the widely applied fusion methods is averaged-based consensus [95, 54, 58, 59, 62, 61]. The consensus algorithm allows each sensor iteratively exchange with neighbours the information obtained from local filtering and perform updates using the information received from its neighbours. With iterations, each node in a network asymptotically converges to the averaged state estimate. For an undirected network, the convergence can be reached if the network is connected, i.e. there exists a path to connect any two arbitrary sensor nodes. Practical sensor networks are often modelled as directional networks where each edge is associated with a direction. A directional network can reach convergence if the network is strongly connected, i.e. there exists a directed path connecting any two arbitrary sensor nodes [96]. For networks with time-varying topology because of node mobility or communication failures (e.g. packet loss), the convergence can be reached if the directional network at each time stamp is strongly connected [97].

Consensus algorithm has been combined with different local filtering techniques. For example, Kalman Consensus Filter (KCF) combines consensus with KF [95] and similarly, Information-weighted Consensus Filter (ICF) combines consensus with IF [58]. KCF exchanges the local target state estimate, i.e. $\mathbf{s}_i^n(t)$, and each camera conducts the average consensus to agree on

the average state estimate at each time step. As the information from each camera is averaged equally, KCF can worsen the overall tracking accuracy due to the presence of cameras with inaccurate state estimates.

To address this problem, ICF computes the average of the state estimate weighted by the covariance matrix of local estimates. To do so, each camera exchanges with neighbours the information form of both local estimate and the corresponding covariance matrix, i.e. $\mathbf{y}_i^n(t)$ and $\mathbf{Y}_i^n(t)$. Let $\mathbf{x}_i^{n,k}(t)$ and $\mathbf{X}_i^{n,k}(t)$ denote the terms that camera c_i exchanges with neighbours for tracking target o_n at each iteration k. The initial terms, i.e. $k = 0$, are computed as [58]:

$$\begin{aligned} \mathbf{x}_i^{n,0}(t) &= \frac{1}{M}\mathbf{y}_i^{n-}(t) + \mathbf{i}_i^n(t) \\ \mathbf{X}_i^{n,0}(t) &= \frac{1}{M}\mathbf{Y}_i^{n-}(t) + \mathbf{I}_i^n(t), \end{aligned} \tag{2.11}$$

where M is the number of cameras in the network. Each camera updates its own terms at k as [58]:

$$\begin{aligned} \mathbf{x}_i^{n,k}(t) &= \mathbf{x}_i^{n,k-1}(t) + \varepsilon \sum_{c_j \in \mathbf{C}_i(t)} \left(\mathbf{x}_j^{n,k-1}(t) - \mathbf{x}_i^{n,k-1}(t) \right) \\ \mathbf{X}_i^{n,k}(t) &= \mathbf{X}_i^{n,k-1}(t) + \varepsilon \sum_{c_j \in \mathbf{C}_i(t)} \left(\mathbf{X}_j^{n,k-1}(t) - \mathbf{X}_i^{n,k-1}(t) \right), \end{aligned} \tag{2.12}$$

where ε is the weight given to the additional information that is contributed by neighbours and $\mathbf{C}_i(t)$ is the set of neighbours of c_i at t. $0 < \varepsilon < 1/D$ is necessary for the convergence where D is the maximum degree of the network [96]. ε affects the convergence speed of the consensus and can be set accordingly [58]. The number of iterations needed to reach consensus in a network depends on the network connectivity, which is often measured by the average degree of the network [96]. The higher connectivity results in a quicker convergence. Extended Information-Weighted Consensus Filter (EIWCF) extends ICF in order to address the non-linearity in visual tracking by replacing the use of IF to EIF [62].

Consensus-based fusion algorithms rely on the parameter, ε that is dependent on the network connectivity. When such knowledge is not available, e.g. the network topology is time-varying, other distributed fusion techniques, such as Iterative Covariance Intersection (ICI) can be applied. ICI computes the weighted average purely based on local state estimate [53, 61]. The initial terms to exchange are the information forms of the local estimate and its corresponding covariance

matrix, i.e.:

$$\begin{aligned} \mathbf{x}_i^{n,0}(t) &= \mathbf{y}_i^n(t) \\ \mathbf{X}_i^{n,0}(t) &= \mathbf{Y}_i^n(t). \end{aligned} \tag{2.13}$$

At each iteration k, the updates are computed as:

$$\begin{aligned} \mathbf{x}_i^{n,k}(t) &= \sum_{c_j \in \mathbf{C}_i(t)} \mu_j^{n,k}(t) \mathbf{x}_j^{n,k-1}(t) \\ \mathbf{X}_i^{n,k}(t) &= \sum_{c_j \in \mathbf{C}_i(t)} \mu_j^{n,k}(t) \mathbf{X}_j^{n,k-1}(t), \end{aligned} \tag{2.14}$$

where $\mu_j^{n,k}(t)$ is the weight assigned to the information from c_j at iteration k.

The weight can be computed with different techniques using either the traces or the determinants of information matrices. For example, the Fast Covariance Intersection (FCI) uses the traces and computes the weight of c_i at iteration k for tracking o_n at t as [53, 93]:

$$\mu_i^{n,k}(t) = \frac{tr(\mathbf{X}_i^{n,k-1}(t))}{tr(\mathbf{X}^{n,k-1}(t))}, \tag{2.15}$$

where $tr(\cdot)$ is the trace of a matrix and $\mathbf{X}^{n,k-1} = \sum_{c_j \in \mathbf{C}_i(t)} tr(\mathbf{X}_j^{n,k-1}(t))$. With ICI, the converged values are not arithmetic averages of the initial states in general [53].

The above mentioned distributed fusion methods consider the case of tracking a single target and involve all cameras in the network. When extending to multi-target tracking, there are extra issues that need to be addressed including the measurement association and the resource limitations. The measurement association is beyond the focus of this thesis but can be possibly addressed with techniques such as the joint probability data association filter [59]. In a camera network, tracking a target with all cameras participating in the fusion process is unnecessary as only those that are close to the target may contribute information for tracking due to limited FoV. When there are resource constraints, such as limited network bandwidth or energy, involving all cameras can be resource-consuming as the number of targets increases and leads to a slow convergence as the network scale increases. It is therefore of great importance to form a subset of cameras dynamically for tracking targets (camera coalition formation) with WSCNs [92].

2.1.3 Coalition formation with resource constraints

This section discusses methods of forming coalitions for tracking multiple targets, where each target is tracked independently and each camera only communicates with neighbours. Methods are classified into three categories based on the number of cameras joining tracking and how the cameras are selected: single camera switching, decentralised coalition and distributed coalition. Single camera switching assigns one camera to track one target at each time [16, 101, 23, 74, 41, 33, 141]. Decentralised coalition selects a subset of cameras for tracking a target at a time with one camera working as a coordinator to select the cameras in the coalition and aggregate the measurements [106, 84, 76, 40, 18, 129, 116]. Finally, distributed coalition involves a subset of cameras for tracking a target at a time, but with each camera independently deciding whether to join in a coalition [61].

In camera switching methods, each camera needs to robustly handover the target information to the next camera [101, 16, 47, 23, 75, 41]. The camera that has a target leaving its FoV can warn its neighbouring cameras to get ready for tracking the target [101] or determine the next camera based on the predicted target motion [15]. A master-slave scheme was used to create an overlapping duration of a master camera (the current tracking camera) and a slave camera (the next selected camera) in order to ensure a robust target information handover in the case of transmission delay [16]. In the case where the network has predefined clusters, the camera with the best view on the target within each cluster can be selected for tracking [47]. Concepts in game theory were also applied for camera handover, where one camera bargains with other cameras in an iterative way in order to assign cameras to targets with the objective to maximise the overall tracking performance [75]. Targets can also be assigned using auction-based methods. A camera considers the targets that it is currently tracking as bids to sell to neighbouring cameras [41]. Each candidate camera makes a sealed bid for a target, and the selling camera will award it to the highest bidder at the second highest price (Vickrey auction).

On-board resource limitations such as energy and computation capability can influence the tracking performance, for example, limited computation capability may lead to low frame rates in real-time tracking systems as the number of tracked objects increases [23]. The resource costs can be incorporated into the price in the auction-based methods [33, 141] or combined with the viewing performance when selecting one camera among cameras that are viewing the same target (viewing cameras) [40]. In general, tracking with one camera is less resource demanding

compared to tracking with multiple cameras, but the disadvantage is the lack of robustness to single camera failures. Multiple viewing cameras can form a group to jointly track a target, while the trade-off is a higher demand in energy and bandwidth as more cameras are involved in processing and communication. Therefore in the literature, both viewing quality/tracking accuracy and limited resources are considered when forming coalitions for multi-camera fusion.

In decentralised coalition methods, one camera works as a local centre for selecting a subset of viewing cameras and aggregating the measurements [106, 129, 18, 116, 13]. The local centre can be dynamically elected, e.g. the camera with the best viewing [75, 47] or most accurate estimate [116]. The local centre can either directly fuse the information from all viewing cameras [18] or further select a subset of viewing cameras with the criteria based on the information contribution, such as cameras with minimum tracking uncertainty [129] and/or the resource cost [106, 116, 13]. ContractNet protocol has been used for forming such coalition for tracking with the constraint that one camera only tracks one target at a time [106]. This is to roughly account for the resource constraints. A more recent work adopts ContractNet protocol but addresses the resource cost via iterative negotiation [116]. Each camera is selected to join the coalition in a greedy manner based on the marginal utility computed with both the information gain and energy cost. In addition to energy cost, the communication performance due to limited bandwidth also affects tracking accuracy. Packet errors has been accounted when selecting a subset of viewing cameras [13], while other factors such as packet loss due to the large network traffic are not taken into account. Above mentioned works are based on the assumption that all viewing cameras are neighbours or a routing table is available. When viewing cameras can not directly communicate with each other, multiple coalitions may be formed for tracking the same target [84], which can lead to more communication and computational resources due to the complicated coalition dynamics.

Each camera can also decide on its own whether to join the coalition for tracking a target instead of being told by a local centre [61]. Distributed coalition often comes with distributed tracking which is performed within the formed coalitions. Distributed tracking avoids multiple coalitions tracking a single target as long as the cameras in a coalition forms a connected subnet, i.e. each camera has a route to any other cameras in the sub-network. Distributed coalition can be formed based on the ratio of the sensing and communication range in order to include all viewing cameras within the coalition [61]. However such way may create large redundancy of

Table 2.1: Summary of the state-of-the-art works for decentralised/distributed tracking with WSCNs. Key. Dec.: Decentralised; Dis.: Distributed; IG: Information Gain; RC: Resources Constraints; CI: Communication Imperfection; ✓: Considered; KF: Kalman Filter; EKF: Extended Kalman Filter; IF: Information Filter; EIF: Extended Information Filter; PF: Particle Filter; KLT: KanadeLucasTomasi feature tracker; AC: Average-based Consensus; ICI: Iterative Covariance Intersection.

Ref.	Coalition			Criteria			Tracking		Target(s)	
	Switch	Dec.	Dis.	IG	RC	CI	Tracker	Fusion	Single	Multiple
[15]	✓			✓			KLT			✓
[101]	✓			✓			EKF		✓	
[47]	✓			✓			KF			✓
[75]	✓			✓			Camshift			✓
[23]	✓			✓			NA			✓
[41]	✓			✓			NA			✓
[33]	✓			✓	✓		NA			✓
[141]	✓			✓	✓		NA			✓
[84]		✓		✓			KF		✓	
[18]		✓		✓			NA		✓	
[106]		✓		✓	✓		NA			✓
[40]		✓		✓			PF		✓	
[116]		✓		✓	✓		IF			✓
[13]		✓		✓	✓	✓	EIF		✓	
[129]		✓		✓			PF			✓
[95]							KF	AC	✓	
[133]							PF	Gossip	✓	
[58]							IF	AC	✓	
[62]							EIF	AC	✓	
[59]							IF	AC		✓
[61]			✓	✓			EIF	ICI	✓	
Proposed			✓	✓		✓	EIF	AC		✓

non-viewing cameras participating into the fusion, which might be acceptable for single target tracking [61]. However when it comes to tracking multiple targets, which is a more realistic scenario, limited bandwidth will deteriorate the tracking accuracy due to poor communication quality. The convergence of distributed fusion can be reached within each formed coalition as long as the sub-network at each time stamp is strongly connected [97].

The detailed state of the art of decentralised/distributed tracking with WSCNs is presented in Table 2.1. The **Proposed** method is referring to the method proposed in Chapter 3.

2.2 Multi-agent active visual tracking

This section covers related works on motion control with respect to the intended multi-agent applications, where each agent independently selects a target and computes its control to actively

track its target while avoiding nearby agents and targets. The section firstly reviews the related works on assigning agents to moving targets, then reviews the methods of controlling an agent for actively tracking a target, and finally discusses the related works on multi-agent collision avoidance.

2.2.1 Assignment of multiple agents to multiple targets

Assigning agents to moving targets is closely related to the CMOMMT problem, which allows each agent to move independently with the objective of maximising the averaged on-target observation time [100, 99, 35, 68, 63, 10]. The agent is formulated with omnidirectional sensing capability and can communicate with neighbouring or all agents [100]. The agent-target assignment is implicitly achieved via local force vectors which are predefined by distance-based functions [100, 99, 35, 68]. The local force vectors encode the attraction forces towards nearby targets and the repelling forces away from nearby agents. CMOMMT focuses on scenarios where there are more targets than agents and an agent is desired to cover multiple targets while avoiding overlapping coverage on a single target. In order to reduce the chances of overlapping coverage, weighted local force vectors are proposed by assigning lower weights on the attraction forces of targets that are within the sensing range of other agents [99]. The weights are further set adaptively under different conditions to achieve higher observation time on targets [35].

With local force vectors, target loss can happen when an agent is covering multiple targets moving in the opposite directions. A *help and tagging* scheme is then introduced to address the target loss problem [68]. An agent that predicts a possible target loss broadcasts a help call to neighbouring agents, providing the target position and the predicted time of target loss. On hearing help calls, agents that are not tagged to any targets estimate the time to capture each candidate target. In order to achieve efficient coverage, each hearing agent performs two rounds of help call selection. The agent first checks the unanswered calls and chooses the closest one among those with the time to capture smaller than the predicted time of target loss. If the hearing agent is not tagged to any help call, it then checks those answered calls and chooses the closest one among those with the target-agent distance smaller than that of the answering agent.

Recent works adapt the original objective of CMOMMT and achieve the agent-target assignment by optimising additional metrics, such as observation fairness [10], observation quality [63] and tracking accuracy [132]. A fair CMOMMT is achieved by maximising the average observation time while minimising the deviation of observation time [10]. The observation quality may

become an issue when using multiple camera-equipped Unmanned Aerial Vehicles (UAVs) as there is a trade-off between the size of observation area and observation quality when flying a UAV at different heights [63]. If the tracking accuracy is of concern, then a balance between the observation time and tracking accuracy is also needed [132]. In general, agents in CMOMMT make independent decisions via neighbourhood communication. However, the neighbours often refer to all agents, i.e. the global knowledge is available to each agent, which is not practical in large-scale applications. CMOMMT treats all targets equally and aims to assign one agent to multiple targets while avoiding one target being observed by multiple agents, which is different from our problem where one agent is assigned to one target for active visual tracking and targets can have different priorities.

2.2.2 Motion control for active visual tracking with a single agent

Active visual tracking with a single agent refers to an agent continuously estimating the state of a target and actively moving towards the target in order to maintain the target appearing centred at its camera's image plane with a certain size. Active visual tracking involves both vision-related processing (e.g. target detection and tracking) and robotic control [142, 130, 77, 48, 91, 102, 49]. Works from the computer vision community and the robotics community tend to address this problem with different focuses. Some works mainly address challenges on visual tracking with simplified motion control methods [119, 67]. Some other works focus on motion control methods but with simplified camera modelling [73, 137] or tracking the target with simplified detections, e.g uniform colour [134, 72, 89] and easily-detected patterns [102]. The mostly used visual tracking techniques are Kanada-Lukas-Tomasi (KLT) feature tracker with clustering [67, 24], Tracking-Learning-Detection (TLD) tracker [102, 49], colour-based detector with temporal filtering [142, 77, 130, 48] and OpenNI tracker for RGB-D sensors [91].

This section focuses on algorithms that computes the robotic controls (controller) for active visual tracking. Feedback controllers are mostly applied for active visual tracking on robotic platforms. Feedback controllers compute the controls that are dependent on the current agent state with the objective of bringing the agent state to be as the same as the desired state [36]. Feedback controllers take as input the difference between the current state and the desired state (the error) and output the robotic control for the current time step. The input error can be either pixel difference on the image plane [77, 102] or distance difference on the ground plane [48, 49]. The frequently used controllers are Proportional-Integral-Derivative (PID) [49] and its variants,

such as P [67, 24], PI [130, 77], and PD [102] controllers. When image-plane errors are used for feedback controllers, there is no need for the knowledge of camera calibration and target size, but a proper parameter tuning is required. When using the ground-plane errors for control, the relative position of the target to the agent on ground plane is required. The relative position can be inferred using a monocular camera with calibration knowledge and target size [102] or using additional on-board sensors, such as laser [67], depth [49, 91] and stereo cameras [24], or using external sensors, such as ceiling-mounted cameras [88].

The controls can also be computed using optimal controllers by minimising a cost function that is composed of state and control variables in order to achieve certain optimality [17]. For example, the controller can be formulated from the tracking perspective by positioning the agent in such a way that the uncertainty of the target state estimate can be minimised by making use of optimal state filters, such as KF [137]. The control can also be computed by minimising a constructed cost function that accounts for a finite time horizon using Model Predictive Control (MPC) [108]. At each time step, MPC computes a set of controls that corresponds to a time horizon and only executes the control corresponding to the current time step. Optimal controllers provide the possibility to account for multiple objectives, such as energy efficiency. Energy cost is often considered when planning the path for an agent with navigation tasks [85, 78, 86], but is seldom considered in computing controls for active target tracking. In addition to using distance as the measure for energy cost, other energy modelling can be used, for example, modelling based on discrete movements (e.g. stops and turns) [86] or modelling based on the kinetic energy [78]. The cost functions are often designed as quadratic functions in order to obtain analytical solutions, i.e. in a feedback form [137], or to compute the solution efficiently for real-time applications [105, 94]. When multiple objectives are considered in the optimisation, the combined cost function becomes more complex and can be designed with variant functions in order to differentiate the penalty strength of multiple criteria. For example, one can use the exponential function to aggravate the penalty on the deviation of a certain criterion [70]. When the cost functions cannot be solved analytically, gradient-based iterative methods are commonly applied to solve the MPC problems compared to the brute-force search method. This is because the brute-force method is more computationally demanding as the solution space increases due to the consideration of multiple time steps. The gradient-based methods are often used with tools, such as *fmincon* in Matlab [90], however the global optimality of the control cannot be

guaranteed.

2.2.3 Collision avoidance in multi-agent scenarios

Obstacles for an agent can be defined as any static or moving objects that are in the way of its desired path. The static obstacles, e.g. a table or a wall [110, 66] can be avoided by using distance-based artificial potential functions [110] or fuzzy logics [66]. In terms of moving obstacles, the literature normally term an object that is moving along a predefined path without reacting to the environment as a passively moving obstacle, such as a preprogrammed vehicle [65, 43, 123, 139], whereas an object that is able to react to the environment dynamics as an actively moving obstacle, such as an agent in a multi-agent scenario [12, 127, 128, 6, 5, 11, 7].

Passively moving obstacles can be avoided with the extension of distance-based methods by incorporating the time-varying property into static obstacles [65]. Another popular method for avoiding passively moving obstacles is by constructing velocity obstacle (VO), which is the set of velocities that could lead to collisions with an obstacle in a time horizon [43]. The agent needs to know the shapes, positions and velocities of itself and nearby passively moving obstacles. The agent then derives the VO induced by each passively moving obstacle assuming that each passively moving obstacle moves at its current velocity within the time horizon. Collision avoidance in the time horizon can be guaranteed if the agent selects a velocity outside the VO. The VO-based method avoids static obstacles by considering static obstacles as passively moving obstacles with zero velocities.

When avoiding agents that can react to the actions of other agents, undesired oscillations can occur if an agent is simply treated as a passively moving obstacle. The motion control of each agent should be collision-free for any other nearby agents and such reciprocity is essential to avoid oscillations among agents. The concept of gyroscopic forces has been combined with distance-based methods to resolve the oscillation issue [20], however its applicability in dense multi-agent scenarios is limited [20]. ORCA is based on VO and has been successfully validated for multi-agent collision avoidance with simulations in densely-packed scenarios [12] and tested on robotic platforms [127, 128, 5, 7, 11]. ORCA defines the set of velocities that are not only reciprocally collision-avoiding but also close to the preferred velocity of each agent. The preferred velocity is the velocity at which an agent would like to move when there is no obstacle in its way. Similar to [43], each agent is assumed to be able to sense the exact shapes, positions and velocities of nearby agents and to infer their preferred velocities without communication. ORCA

provides a sufficient condition for multiple agents to avoid collisions among each another [12]. However, the collision-free velocity may be infeasible in medium or highly dense cases because the set of collision-free velocities is empty. To address this problem, ORCA allows each agent to achieve a feasible velocity by minimally shifting the constraints induced by other agents using a 3D linear program, where the 3D linear program is always feasible [12].

ORCA was originally proposed for holonomic agents, however most of the off-the-shelf robotic platforms follow non-holonomic kinematics, such as differential-drive models [128, 6] and car-like models [5]. In order to make ORCA applicable to real robotic platforms, derivative works extend ORCA to cope with non-holonomic kinematics from two directions [128, 6, 5, 11]. One is to directly compute the robotic control constraints rather than the velocity constraints. The generalised velocity obstacle (GVO) samples the space of accessible controls and determines whether a collision can occur by estimating the time when the agent-obstacle distance achieves the minimum [139]. While GVO only addresses the avoidance of passively moving obstacles, a recent work extends GVO to reciprocal collision avoidance by approximating the control space with polygons and the non-linear control constraints with the first-order Taylor approximation [11]. Another direction follows the original formulation of ORCA and maps the linear collision-avoiding velocity to non-linear robotic control in the end. As there will be trajectory tracking errors caused by the linear formulation, the agent radius is often enlarged when deriving the velocity constraints to compensate the trajectory tracking errors [128, 6, 5].

Each agent needs to track itself and nearby agents, which is often achieved with centralised visual tracking systems in real robotic implementation due to limited on-board sensing capability of the off-the-shelf robotic platforms [127, 128, 6, 11]. One machine tracks all agents using either a single top-view camera [127] or a motion capture system (Vicon) [11], computes the controls for all agents and sends the controls back to each agent for execution. Such centralised frameworks lack the scalability in wide areas. On-board sensors, e.g. IMUs and laser scanners, can be used for each agent to track itself [51]. Each agent exchanges with other agents its own state to enable the on-board computation of collision-free robotic controls.

The state of the art of multi-agent active tracking in presented in Table 2.2. The **Proposed** method is referring to the method proposed in Chapter 5.

Table 2.2: Summary of the state-of-the-art works in multi-agent active tracking. Key. N: Navigation; O: Observation; T: Tracking; CA: Collision Avoidance; #R: The number of robots (agents) assigned to a target; #T: The number of targets an agent is assigned to; OS: Onboard Sensors; ES: External Sensors; S: Capability to avoid static obstacles; P: Capability to avoid passively moving obstacles; A: Capability to avoid actively moving obstacles; ✓: considered/used; M: Multiple.

Ref	Motion objectives				Assignment		Localisation		Obstacles			Robot
	N	O	T	CA	#R	#T	OS	ES	S	P	A	
[100]		✓		✓	1	M	✓		✓			Nomad 200
[99]		✓		✓	1	M	✓		✓			Nomad 200
[35]		✓			1	M						Simulation
[68]		✓			1	M						Simulation
[63]		✓			1	M						Simulation
[10]		✓			1	M						Simulation
[132]		✓	✓		1	M	✓	✓				MikroKopter
[88]			✓		1	1		✓				Pioneer P2-DX
[142]			✓	✓	1	1	✓		✓			ApriAttenda
[67]			✓		1	1	✓					Segway RMP
[24]			✓		1	1	✓					Pioneer P3-AT
[49]			✓		1	1	✓					iRobot Create
[130]			✓		1	1						X4-flyer
[77]			✓		1	1	✓					Helicoper
[37]			✓	✓	1	1	✓		✓			robuLAB10
[48]			✓	✓	1	1	✓		✓			Kompai
[102]			✓		1	1	✓					AR drone
[91]			✓		1	1	✓					AscTec Pelican
[137]			✓		1	1						Simulation
[32]			✓	✓	1	1	✓		✓			Wheeled robot
[12]	✓			✓					✓	✓	✓	Simulation
[127]	✓			✓				✓	✓	✓	✓	iRobot Create
[128]	✓			✓				✓	✓	✓	✓	iRobot Create
[6]	✓			✓				✓	✓	✓	✓	Epuck
[5]	✓			✓					✓	✓	✓	Simulation
[51]	✓			✓			✓		✓	✓	✓	iRobot Create
[11]	✓			✓				✓	✓	✓	✓	iRobot Create
[7]	✓			✓				✓	✓	✓	✓	AR Drone
Proposed		✓	✓	✓	1	1		✓	✓	✓	✓	Simulation

2.3 Summary

In summary, decentralised/distributed tracking frameworks are suitable for tracking with WSCNs. Decentralised tracking schemes require a local centre to form a coalition for each target and perform information fusion [106, 33, 116, 40, 129, 13]. Distributed tracking instead enables each camera with an agreed target state estimate via iterative neighbourhood communication without local centres [95, 54, 58, 59, 62, 34]. Traditional distributed fusion techniques involve all cameras to exchange information with neighbours, which is bandwidth-demanding and unnecessary in tracking applications due to the localised nature. A subset of cameras can be formed into a connected coalition to track a single target [61]. In order to avoid redundant cameras joining in coalitions especially when tracking multiple targets in a bandwidth-limited WSCN, the distributed coalition formation should account for both the information gain for local tracking and the information loss due to the poor link quality [C1].

Agents can track nearby targets by performing distributed fusion with nearby static cameras. On hearing requests from static cameras, agents select their own target. The assignment of multiple agents to multiple targets is closely related to the CMOMMT problem [100, 99, 35, 68] that aims at maximising the on-target observation time. CMOMMT achieves a one-to-multiple agent-target assignment, which is different from our intended applications with a one-to-one agent-target assignment. Targets in the intended applications may have different tracking priorities, therefore a prioritised agent-target assignment scheme is needed to improve the observation time on targets of a higher priority while avoiding overlapping coverage. Once a target is assigned, the agent computes the control to perform active visual tracking. Although feedback-based controllers [49, 67, 24, 130, 77, 102] are mostly used in practical applications, minimisation-based controllers can be exploited for multi-objective control [C2].

Collision avoidance among agents is an important problem to tackle, which has been addressed by the ORCA method and has been successfully validated in multi-agent navigation applications [12, 127, 128, 5, 11]. The path of an agent performing a navigation task is rather flexible during avoidance manoeuvres as long as the goal position is reached. However for agents performing active visual tracking, the paths are constrained by target dynamics and limited FoV. Agents can lose their target out of the FoV when avoiding nearby agents and targets. View maintenance should be accounted when applying ORCA in multi-agent active visual tracking [C4].

Tracking is important for multi-agent systems. Due to the limited on-board sensing capa-

bilities, tracking is often achieved through a centralised vision system, for example using a top-down camera [127, 128, 6] or a motion capture system [11]. However, centralised systems are not scalable and distributed approaches are preferable to favour the scalability and flexibility of deployment. Distributed approaches exist where agents exploit their IMUs and laser scanners for self localisation and exchange with each other the estimated states for on-board control computation [51]. However, the approach in [51] is not able to track targets in crowded scenes. The limited on-board sensing also constrains the set of possible collision-free controls within the visibility range and reduce the feasible control space. The exploitation of ambient cameras and distributed tracking techniques help to achieve both tracking accuracy and system scalability [C5].

Chapter 3

Coalition formation for distributed tracking with static cameras

---

3.1 Overview

This Chapter focuses on the collaboration among static cameras that continuously perform distributed tracking on targets. The collaboration defines what information is exchanged among static cameras, how the information is exchanged and how the information is used for local decision making. A communication-aware coalition formation scheme prior to distributed tracking is proposed to account for imperfect communication performance caused by limited bandwidth [C1]. This Chapter is organised as follows. Section 3.2 describes the proposed communication-aware coalition formation scheme for distributed target tracking. Section 3.3 evaluates the coalition formation scheme with simulations in terms of tracking accuracy and communication cost. Finally, Section 3.4 summarises this chapter.

3.2 Proposed method

Static cameras perform target detection (e.g. using [38]) and local tracking with EIF. We assume that i) the on-board computational resources of each static camera can afford real-time target detection and tracking, ii) the detection association within and across camera views is perfect, iii) each static camera has the knowledge of its neighbouring cameras, i.e. who and where are the neighbours.

Let $\mathbf{C}_n(t)$ be the set of cameras that are in the coalition for tracking target o_n at t. Each

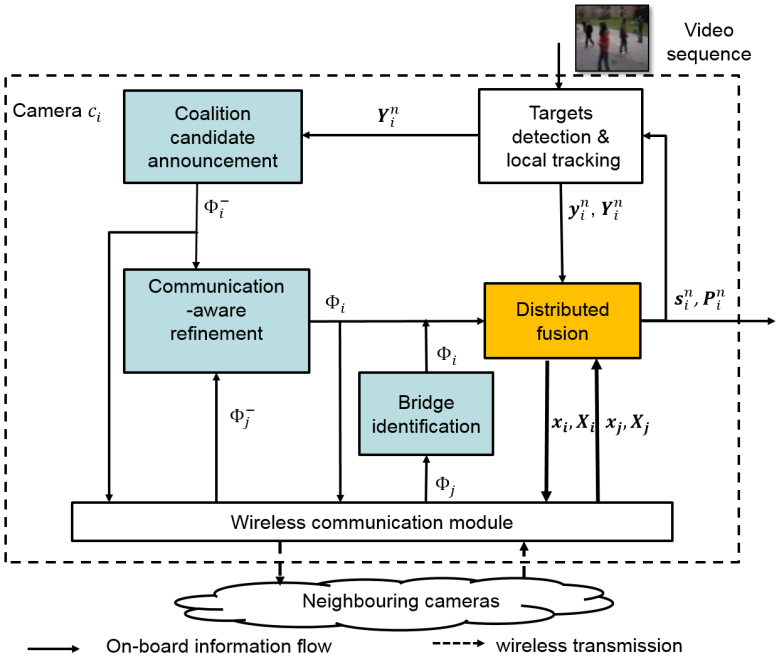


Figure 3.1: The block diagram of the communication-aware coalition formation for distributed tracking. The blocks in blue are proposed ones, the block in orange uses state-of-the-art methods and the block in white is beyond the scope of this thesis. The arrows indicate the information flow where the thick ones represent iterative information exchanges.

camera c_i is allowed to join multiple tracking coalitions at a time, and $\Phi_i(t)$ is the set of tracking coalitions that c_i joins in at t. Each camera decides its own coalition set $\Phi_i(t)$ with the objective of improving tracking accuracy under limited bandwidth. Distributed tracking is then performed at all cameras that are within a tracking coalition.

We propose a coalition formation scheme via neighbourhood messaging. The scheme is composed of three stages, namely, coalition candidate announcement, communication-aware coalition refinement and bridge identification (see Fig. 3.1). The first stage includes all cameras with information that can be contributed to tracking a target. The second stage further refines the cameras by considering the marginal information gain under imperfect communication link quality. The final stage avoids forming multiple coalitions for tracking a single target, especially in cases when cameras have information of the same target but cannot directly communicate with each other (see Fig. 3.2(b)). The resulted tracking coalition forms a connected subset of cameras that contains most target information under realistic communication modelling at each t. We omit t in the following sections to simplify the notations.

Coalition candidate announcement

Each camera c_i firstly decides its candidate coalition set, Φ_i^- , and announces it to its neigh-

bours. c_i becomes a candidate in the coalition for tracking o_n if c_i has information to contribute that is of sufficient tracking confidence, i.e. either with a current or a previous measurement of the target. Once c^i decides its candidate coalitions, it broadcasts a candidate message, m_i^C (superscript C for 'Candidate'), to its neighbours. m_i^C contains Φ_i^- and the corresponding information matrices of local target state estimates. After this round of information exchange, each camera has knowledge of the candidate sets of its neighbours together with their tracking confidence of each candidate target.

Communication-aware coalition refinement

The candidate set Φ_i^- is further refined by the communication performance using a marginal information utility. On one hand, when a camera joins in a tracking coalition for a target, the camera contributes not only its own information to the state estimation but also increases the neighbourhood traffic, thus resulting in a worse link quality. On the other hand, when a camera does not join in the tracking coalition, it contributes neither the information nor the traffic to its neighbourhood. Therefore the actual information utility that each camera contributes to the coalition combines both the tracking information utility and the link quality.

Let U_i^n be the information utility that c_i contributes to tracking o_n. U_i^n can be computed as the determinant of the information matrix corresponding to local target state estimate:

$$U_i^n = \det(\mathbf{Y}_i^n), \tag{3.1}$$

where $\det(\cdot)$ is the determinant of a matrix. The larger value indicates more certainty of the state estimate.

The link quality is measured by the Packet Reception Ratio (PRR), which is defined as the ratio of successfully received packets to all sent packets. The link quality is modelled as a Bernoulli random process [60] by taking into account both the Packet Error Ratio (PER) and the Packet Loss Ratio (PLR), assuming that there is no channel interference. PER defines the ratio of erroneously received packets to all sent packets, and PLR defines the ratio of not received packets to all sent packets.

The PER, ρ^e, is related to the transmission distance d and it is commonly approximated by

a sigmoid function [13]:

$$\rho^e = \frac{\mathfrak{b}}{1 + e^{\mathfrak{a}\left(\frac{1}{d^2} - \mathfrak{c}\right)}}, \tag{3.2}$$

where $\mathfrak{a}$, $\mathfrak{b}$ and $\mathfrak{c}$ are protocol-dependent constants that can be fitted with experimental data [13]. The PLR, ρ^l, is defined as:

$$\rho^l = 1 - \frac{P^R}{P^S}, \tag{3.3}$$

where P^R is the number of packets received in the network per second and P^S is the number of packets sent per second.

We approximate P^R as the network throughput, which is the rate of information received (in bytes) in a network. Given a wireless network with a random topology, the throughput, $\mathfrak{W}$, can be modelled as [50]:

$$\mathfrak{W}(m) = \Theta\left(\frac{\mathfrak{C}}{\sqrt{m \log m}}\right), \tag{3.4}$$

where m is the number of nodes that are competing to transmit and $\mathfrak{C}$ is the network capacity. $\Theta(\cdot)$ is the model function [50]. We approximate P^S as the rate of neighbourhood traffic (in bytes) generated during the distributed fusion phase. If K iterations are performed during distributed fusion, the neighbourhood traffic of c_i, P_i^S, that accounts for the bytes sent by both c_i and its neighbours $\mathbf{C}_i$ can be approximated as:

$$P_i^S = \frac{1}{\Delta T^F} \sum_{k=1}^{K} \left(\sum_{c_j \in \mathbf{C}_i} L_j^k |\Phi_j| + L_i^k |\Phi_i| \right), \tag{3.5}$$

where ΔT^F is the time duration of distributed fusion and L_i^k is the packet load of c_i at each iteration k and $|\cdot|$ is the cardinality of a set.

Let ρ_i^l be the PLR of c_i:

$$\rho_i^l = 1 - \frac{\mathfrak{M}\mathfrak{C}}{P_i^S \sqrt{|\mathbf{C}_i| \log |\mathbf{C}_i|}}, \tag{3.6}$$

where $\mathfrak{M}$ is a protocol-dependent constant.

Let ρ_{ij} be the link quality, i.e. PRR, between camera c_i and c_j and let ρ_{ij}^e be the packet error

ratio between c_i and c_j. We compute ρ_{ij} as:

$$\rho_{ij} = 1 - \rho_i^l - \rho_{ij}^e, \tag{3.7}$$

Let $\mathbf{C}_i^n$ be the set of neighbouring cameras of c_i that are in the same coalition for tracking o_n, i.e. $\mathbf{C}_i^n = \mathbf{C}_i \cap \mathbf{C}_n$. Let U_i^{n+} be the actual information utility that c_i contributes to $\mathbf{C}_i^n$ by joining the coalition, while U_i^{n-} be the actual information utility that c_i contributes to $\mathbf{C}_i^n$ by not joining $\mathbf{C}_n$. We can compute U_i^{n+} and U_i^{n-} as:

$$\begin{aligned} U_i^{n+} &= U_i^n \sum_{j \in \mathbf{C}_i^n} \rho_{ij}^{n+}, \\ U_i^{n-} &= U^n \sum_{j \in \mathbf{C}_i^n} \left(\rho_{ij}^{n-} - \rho_{ij}^{n+} \right), \end{aligned} \tag{3.8}$$

where $U^n = \sum_{c_j \in \mathbf{C}_i^n} U_j^n$. ρ_{ij}^{n+} and ρ_{ij}^{n-} are the estimated link quality between c_i and one of its neighbour c_j based on the neighbourhood traffic including and excluding c_i joining in $\mathbf{C}_n$, respectively. ρ_{ij}^{n+} and ρ_{ij}^{n-} are computed using the worst-case neighbourhood traffic, i.e. all neighbours join in their candidate set Φ_j^- for distributed tracking.

Let ΔU_i^n be the marginal information utility that camera c_i contributes to track o_n with the consideration of link quality:

$$\Delta U_i^n = U_i^{n+} - U_i^{n-}. \tag{3.9}$$

If $\Delta U_i^n \geq 0$, then c_i joins in $\mathbf{C}_n$ and updates its coalition set Φ_i. At the end of this stage, each camera broadcasts a member message, m_i^M (superscript M for 'Member'), that contains Φ_i to neighbours.

Bridge identification

Due to the limited communication range, cameras that join in the same coalition may not form a connected sub-network, however such connectivity is required to achieve convergence in distributed fusion. We therefore design the cameras that are not in the coalition to identify themselves as bridge cameras based on the reception of messages. A member camera that receives other member messages confirms with neighbours by broadcasting an acknowledgement message, m_i^A (superscript A for 'Acknowledgement'). A camera that is not in a coalition joins the coalition as a bridge when this camera receives more than one

member messages and receives less acknowledgement messages than member messages. The bridge camera broadcasts a bridge message, m_i^B (superscript B for 'Bridge'), to neighbours, which contains the information from all received member messages. It can happen that a bridge camera in $\mathbf{C}_n$ does not have any prior information for tracking o_n. We therefore initialise the local tracking of the bridge camera using the averaged information that is received from $\mathbf{C}_i^n$ at the candidate announcement stage.

At the end of the coalition formation, a connected subset of cameras with most accurate information to contribute under imperfect link quality is formed for performing distributed tracking. One example of the message sequence diagram is shown in Fig. 3.2. Note that if the network bandwidth is sufficient, i.e. lossless and delay-free communication links, only the first and third stages are required for coalition formation.

Distributed tracking techniques, such as EIWCF [62] can be employed within each formed coalition if the network connectivity is known. Algorithm 1 shows the distributed tracking performed within the coalition for tracking target o_n. Any cameras that are not inside $\mathbf{C}_n$ but occupy related information of target o_n perform local tracking and may join in $\mathbf{C}_n$ at a future time step.

Algorithm 1 Distributed tracking algorithm that runs on c_i for tracking target o_n

Input:

$\mathbf{C}_i^n$: Neighbouring cameras of c_i in the coalition for tracking o_n
$\mathbf{z}_i^n$: Measurement of c_i corresponding to target o_n
$\mathbf{R}_i^n$: Covariance matrix of $\mathbf{z}_i^n$
K : Iterations of fusion

compute the measurement information $\mathbf{i}_i^n$ and $\mathbf{I}_i^n$ (Eq. 2.7)
compute the prior state estimate information $\mathbf{y}_i^{n-}$ and $\mathbf{Y}_i^{n-}$ (Eq. 2.5)
compute the state estimate information $\mathbf{y}_i^n$ and $\mathbf{Y}_i^n$ (Eq. 2.8)

prepare the initial terms $\mathbf{x}_i^{n,0}$ and $\mathbf{X}_i^{n,0}$ using Eq. 2.11 with $M = |\mathbf{C}_i^n|$
broadcast $\mathbf{x}_i^{n,0}$ and $\mathbf{X}_i^{n,0}$ to $\mathbf{C}_i^n$

for $k < K$ **do**
  receive $\mathbf{x}_j^{n,k-1}$ and $\mathbf{X}_j^{n,k-1}$, $\forall c_j \in \mathbf{C}_i^n$
  update $\mathbf{x}_j^{n,k}$ and $\mathbf{X}_j^{n,k}$ using Eq. 2.12 with $\mathbf{C} = \mathbf{C}_i^n$
  broadcast $\mathbf{x}_j^{n,k}$ and $\mathbf{X}_j^{n,k}$
end for

compute the target state estimate $\mathbf{s}_i^n$ and $\mathbf{P}_i^n$ based on Eq. 2.4

Output: $\mathbf{s}_i^n$ and $\mathbf{P}_i^n$

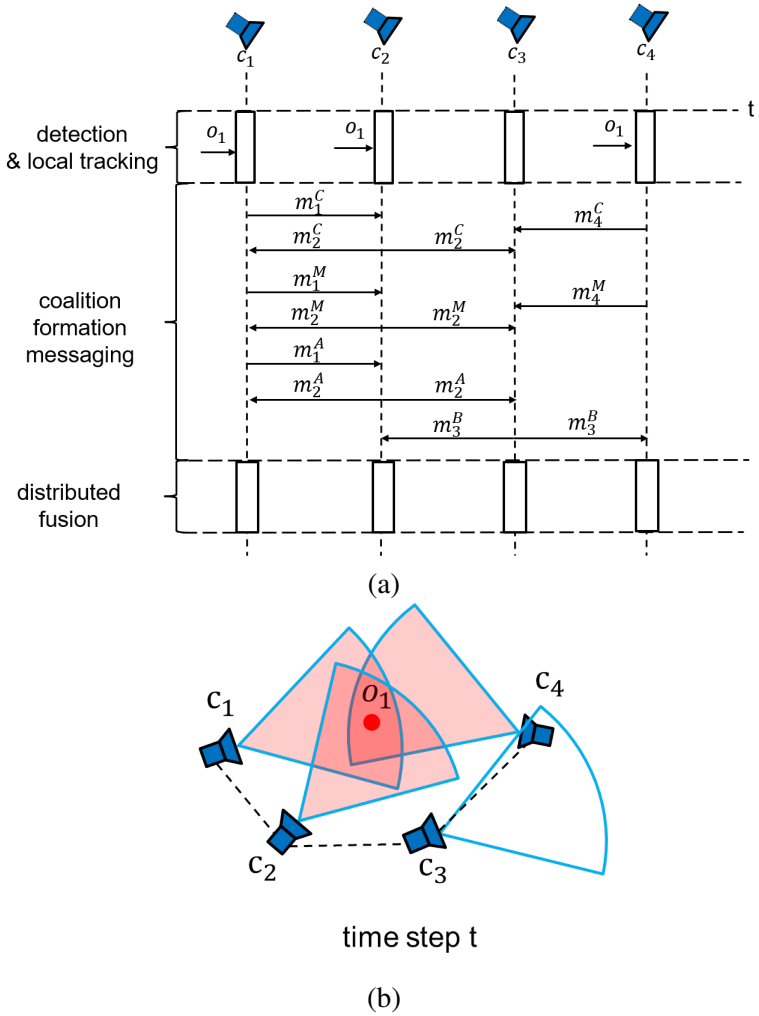


Figure 3.2: Sequence diagram of static-camera coalition formation for tracking one target. (a) The sequence diagram shows the messaging among static cameras in one time step. (b) The corresponding scene where four static cameras track target o_1 at t. A FoV is coloured by the target colour when the target is within the FoV.

3.3 Validation

This section validates the proposed framework with simulations. We validate the coalition formation among static cameras in terms of tracking accuracy and communication efficiency, and compare it with decentralised and centralised schemes.

In order to demonstrate the impacts of the communication-aware refinement stage, we test the proposed coalition formation strategy in two modes, a partial formation mode (EIWCF-PC), i.e. coalition candidate announcement and bridge identification, and a full coalition formation process (EIWCF-FC) with all three stages. We evaluate the tracking accuracy and communication cost with varying iterations K and the number of targets N, and compare it with (a) EIWCF with-

out coalition formation [62] and (b) a decentralised strategy (DECEIF) [116]. DECEIF forms a coalition of a subset of cameras that can observe a target (viewing cameras) with one camera aggregating the measurements for tracking using EIF. The Centralised Extended Information Filter (CEIF) is used as a baseline for tracking accuracy comparison [62]. CEIF uses the measurements from all viewing cameras to estimate the target state. We also perform CEIF using cameras that are within the coalitions formed by our full coalition formation (CEIF-FC) in order to compare the converged state estimate within formed coalitions to the averaged state estimate.

We first evaluate the fusion convergence, tracking accuracy with the coalition formation by increasing iterations K under two types of network. Communication is ideal in this set of experiments, i.e. there is no packet loss/error over a communication link. We then demonstrate the tracking accuracy achieved by the coalition formation scheme under realistic communication performance modelling. We quantify the tracking accuracy as the mean tracking error (and its standard deviation) between the estimated and ground-truth position. The mean tracking error is defined as:

$$\frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N}\frac{\sum_{c_i\in\mathbf{C}_n(t)}\left\|\mathbf{s}_n^i(t)-\mathbf{s}_n(t)\right\|}{\|\mathbf{C}_n(t)\|}, \tag{3.10}$$

where $\mathbf{s}_n(t)$ is the corresponding ground-truth state, N is the number of targets, and T is the whole experiment duration. $\|\mathbf{C}_n(t)\|$ means the cardinality of the set $\mathbf{C}_n(t)$.

The communication cost is approximated by the number of transmissions assuming that one packet only contains the information related to one target.

3.3.1 Experiment setup

Let there be 30 cameras with overlapping FoVs monitoring a 100m $\times$ 100m scene. Each camera has view angle of 140° and is randomly placed in a grid manner at height of 5 m with a top-down square FoV. Targets move independently within the scene following a non-linear motion model and measurement model as in [62] (Fig. 3.3(a)). We set the maximum target speed to 2 m/s to simulate a normal walking speed. We model the measurement covariance to be proportional to the target-camera distance with an upper-bound $\mathbf{R}_{\max} = diag([2\ 2])$.

Two types of network with different network connectivities are used in the experiments. The Type I network has a smaller connectivity, where not all cameras with overlapping FoVs are connected (see Fig. 3.3(b)). The Type II network has a larger connectivity where all cameras with

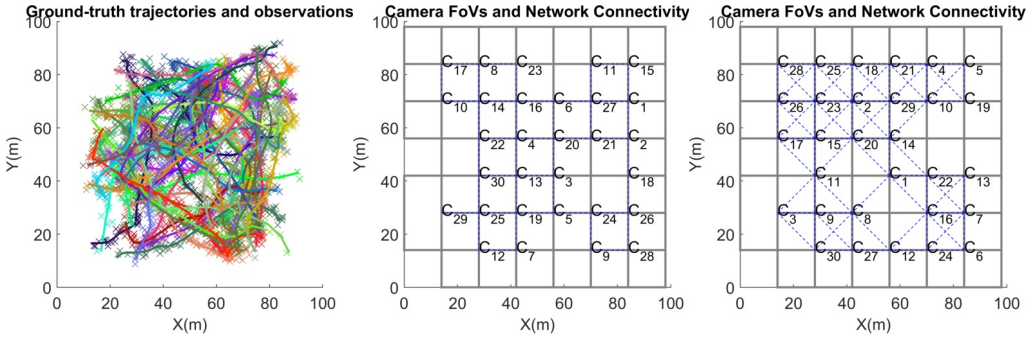


Figure 3.3: Experimental setup. (a) Simulated trajectories and observations on the ground plane of 50 targets for 25 time steps. (b) A Type I network. (c) a Type II network. Each camera is placed in the centre of a square FoV with grey outline. The connectivity between cameras is indicated by a blue dashed line.

overlapping FoVs are able to communicate with each other (see Fig. 3.3(c)). We achieve different connectivities by varying the communication range. The decentralised strategy DECEIF is tested only with a Type II network as the method assumes that all viewing cameras are neighbours. We set the parameters for the communication model based on existing experimental studies [145, 103] assuming the use of standard IEEE 802.15.4 protocol. $\mathfrak{M}$ is set to 0.85 in Eq. 3.6 and the ρ^e-related parameters in Eq. 3.2 are set to $[\mathfrak{b},\ \mathfrak{a},\ \mathfrak{c}] = [0.1,\ 1.5,\ 0.1]$. The neighbourhood traffic considers mainly the communication load during the distributed tracking phase, which is more demanding compared to the coalition formation phase. Each camera transmits the same amount of data (the local estimated information vector and corresponding information matrix) during the fusion within its coalition at each iteration [62]. The communication load L_i^k is the same for all nodes (around 75 bytes). We set ΔT^F to 0.1 s for real-time tracking consideration.

3.3.2 Results discussion

Fig. 3.4 shows the results of mean tracking error when tracking 10 and 30 targets with increasing K with two types of network under ideal communication. Distributed tracking within coalitions (EIWCF-PC and EIWCF-FC) converges much quicker than EIWCF due to a smaller sub-network scale. We define the convergence speed as the iteration when the difference of mean tracking error between two consecutive iterations occurs to be less than 0.1. Without coalition formation, EIWCF converges at the eighth and seventh iteration with a Type I and a Type II networks, respectively, while the convergence speeds of EIWCF-FC are three and one under a Type I and a Type II network, respectively. EIWCF-FC converges after one iteration in a Type II network as

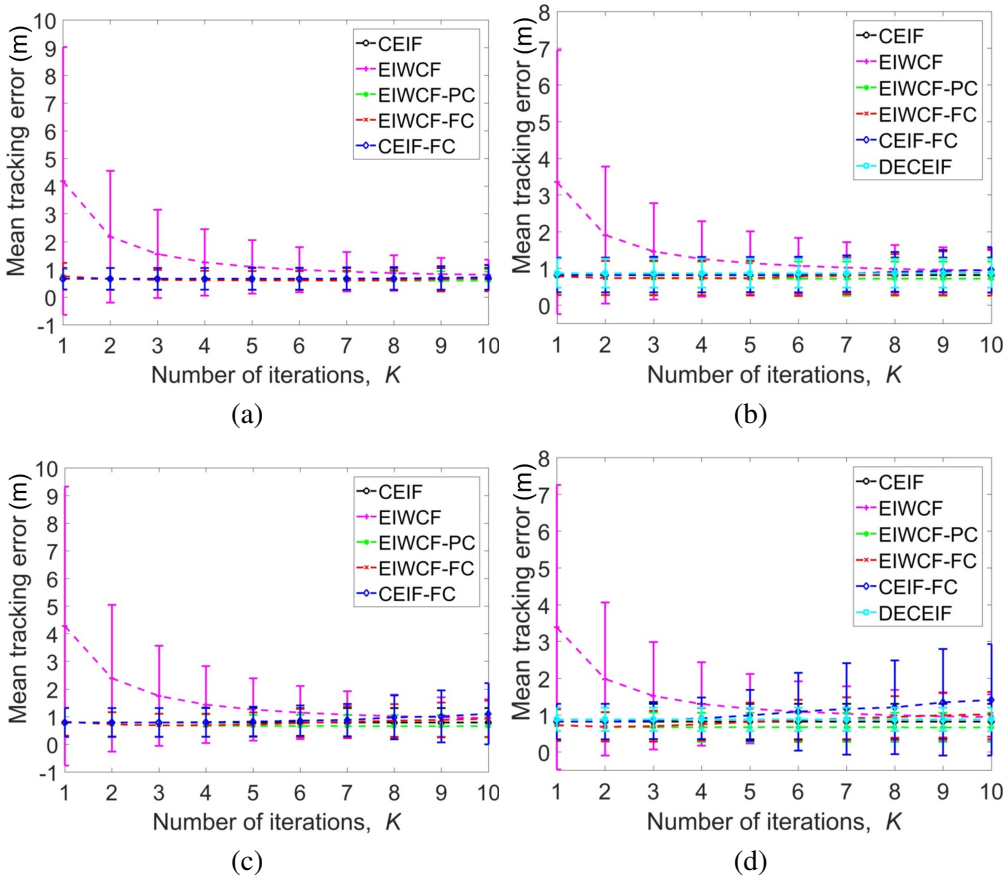


Figure 3.4: Mean tracking error with increasing numbers of iterations ($K \in [1, 10]$) without packet error and packet loss. (a, b) Results of tracking 10 targets with a Type I network and a Type II network, respectively. (c, d) Results of tracking 30 targets with a Type I network and a Type II network, respectively.

the formed coalitions are fully connected.

Compared to the mean tracking error achieved by DECEIF, partial coalition formation, i.e. EIWCF-PC, can achieve a smaller tracking error after convergence in a Type II network. This is because EIWCF-PC includes all cameras with information to contribute to tracking, e.g. current viewing cameras together with previously viewing cameras while CEIF only involves current viewing cameras. Interestingly, we also notice that the mean tracking error of EIWCF-FC may increase as the number of iterations increases. This phenomenon is more apparent when there are more targets to track in a network with a larger connectivity as shown in Fig. 3.4(d). This is because more targets and a larger network connectivity lead to high volume of neighbourhood traffic. The volume of neighbourhood traffic becomes even higher as the number of iterations increases, which makes many cameras quit their candidate coalitions, thus leading to less accurate tracking. The increasing trend of the mean tracking error achieved by CEIF-FC in Fig. 3.4(d)

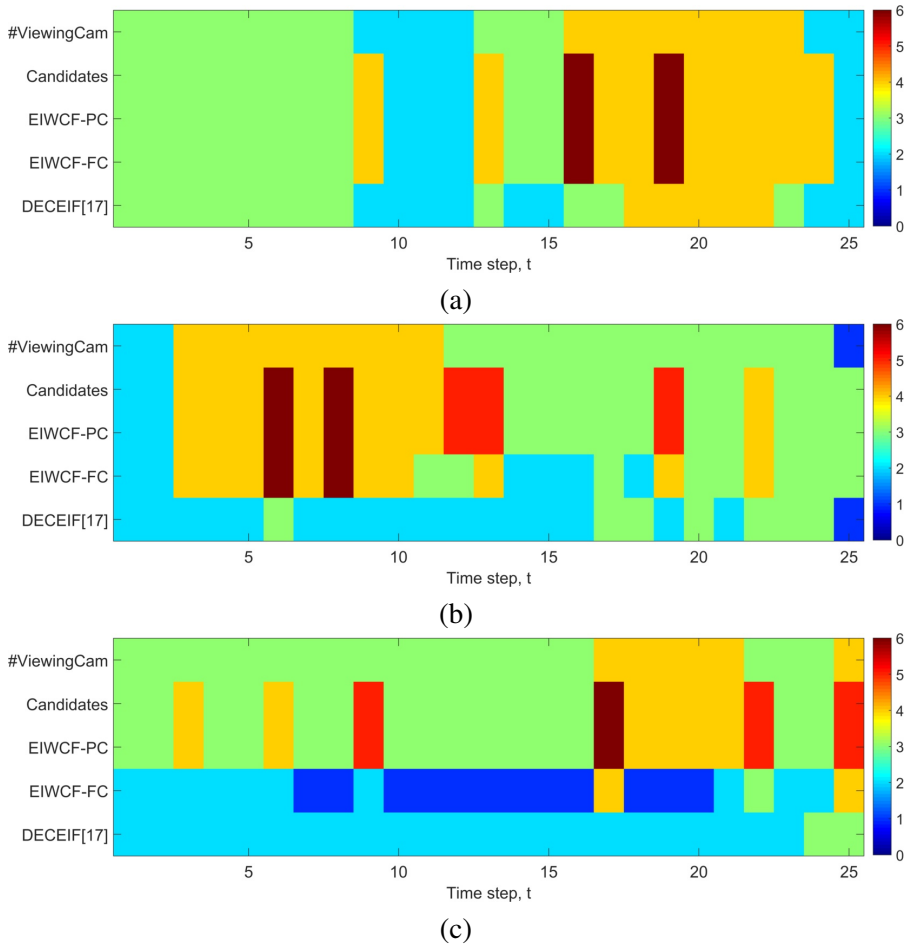


Figure 3.5: The number of cameras in a coalition for tracking one target over time in a Type II network. (a) 15 targets (half of the number of cameras) (b) 30 targets (same to the number of cameras) and (c) 60 targets (double of the number of cameras).

also explains this phenomenon. EIWCF-FC at the convergence iteration achieves a smaller mean tracking error than that of DECEIF, as the proposed coalition considers all cameras with information that can be contributed to tracking, whereas DECEIF considers only a subset of viewing cameras.

In the following sets of experiments, we set the iteration number to one in a Type II network. Fig. 3.5 shows the coalition size for one of the targets over time in a Type II network with 15, 30 and 60 targets. Fig. 3.5 shows the number of viewing cameras (#ViewingCam), the number of candidate cameras at the first stage (Candidates), the formed coalition size with partial coalition (EIWCF-PC), the formed coalition size with full coalition (EIWCF-FC) and the formed coalition size using a decentralised strategy (DECEIF) [116]. The number of candidate cameras is always larger or equal to the number of viewing cameras as we also include cameras with measurements at the previous time step. When the target density is low (Fig. 3.5(a)), EIWCF-PC and EIWCF-

FC include all candidate cameras into the coalitions. As the target density increases, especially in the middle of the experiments when most targets are intersecting, EIWCF-FC includes only partial candidate cameras due the limited communication performance. The decentralised strategy forms coalitions with the size smaller than the number of viewing cameras as the method selects a subset of cameras among the viewing cameras. The average percentage of candidate members in the coalition formed with EIWCF-FC under the three target densities is 99%, 95% and 72%, respectively, and the average percentage of viewing cameras in the coalition formed with DECEIF is 90%, 79% and 70%, respectively. The proposed coalition becomes more responsive in terms of reducing the size of coalitions when the target density is large, e.g. when the number of targets is double of the number of cameras. With the proposed coalition formation, it can happen that all cameras quit to join in a coalition for tracking a target and perform only local tracking.

Fig. 3.6 shows the mean tracking error with increasing numbers of targets in a Type II network when the packet loss and packet error are modelled. Note that all centralised methods including CEIF and CEIF-FC do not consider any link modelling and only serve as baseline comparisons. EIWCF without coalition formation has much larger mean tracking error compared to other distributed or decentralised methods as shown in Fig. 3.6(a). This is because without coalition distributed tracking generates a high volume of network traffic. Such large amount of traffic leads to lossy links at each time step and accumulates tracking errors over time. Fig. 3.7 shows the mean tracking error when tracking 10 targets over time in a Type II network. The accumulation of tracking error resulted by EIWCF is clear.

Zoom-in results of mean tracking error without EIWCF is shown in Fig. 3.6(b). In general, all distributed or decentralised methods generate a larger mean tracking error as the number of targets increases. The proposed EIWCF-FC for distributed tracking may achieve a slightly worse tracking accuracy compared to that of the DECEIF method. However this is understandable as decentralised tracking requires unicast communication, i.e. each camera in the coalition sends packets to its local centre whereas distributed tracking requires broadcast communication where all cameras within a coalition communicates with each other. The neighbourhood traffic generated by distributed tracking is larger than that of decentralised tracking, which make the tracking accuracy of EIWCF-FC inferior to that of DECEIF. The trade-off is that distributed tracking enables each camera in a coalition with a fused target state estimate while in decentralised tracking only the local centre has such knowledge.

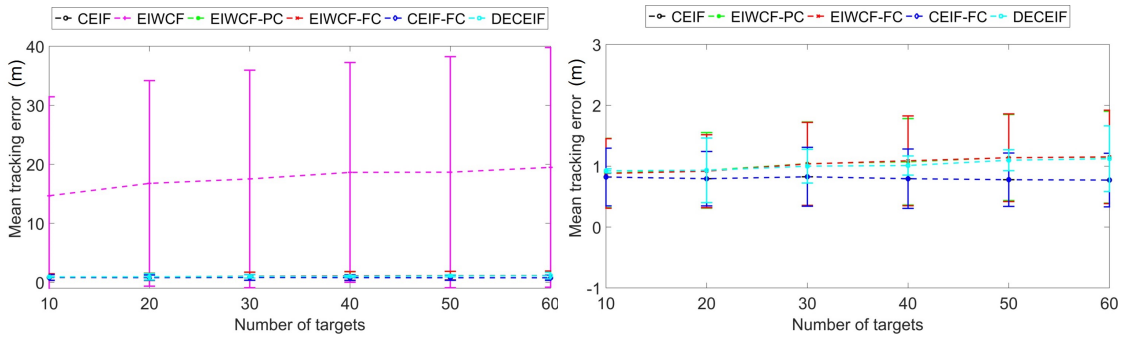


Figure 3.6: Mean tracking error in a Type II network with increasing numbers of targets when the packet error and packet loss are modelled. (a) Results of all methods. (b) Zoom-in results of all methods without EIWCF.

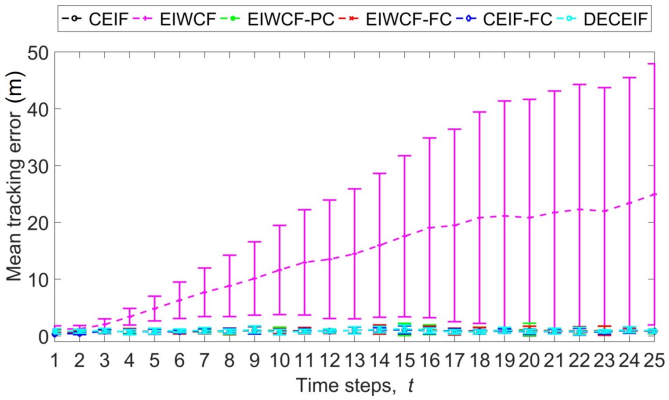


Figure 3.7: Mean tracking error of all methods over time in a Type II network with 10 targets when the packet error and packet loss are modelled.

Fig. 3.8 shows the communication cost of the distributed/decentralised methods with increasing numbers of targets in a Type II network (result of one run). The number of transmissions during the coalition formation and distributed fusion phases is averaged by the number of cameras and experimental time steps. The communication cost of EIWCF-FC with one iteration is comparable to that of DECEIF, but much smaller than that of EIWCF (see Fig. 3.8(a)). The communication cost of DECEIF is contributed by the header election, iterative coalition member selection and measurements aggregation. We further break down the transmissions in EIWCF-FC into different messages types, i.e. m^C (Candidate message), m^M (Member message), m^A (Acknowledgement message), m^B (Bridge message) and m^F (Fusion message). As shown in Fig. 3.8(b), no bridge message is transmitted as viewing cameras in a Type II network are connected. The number of Fusion messages is similar to that of other types of messages because only one iteration is performed.

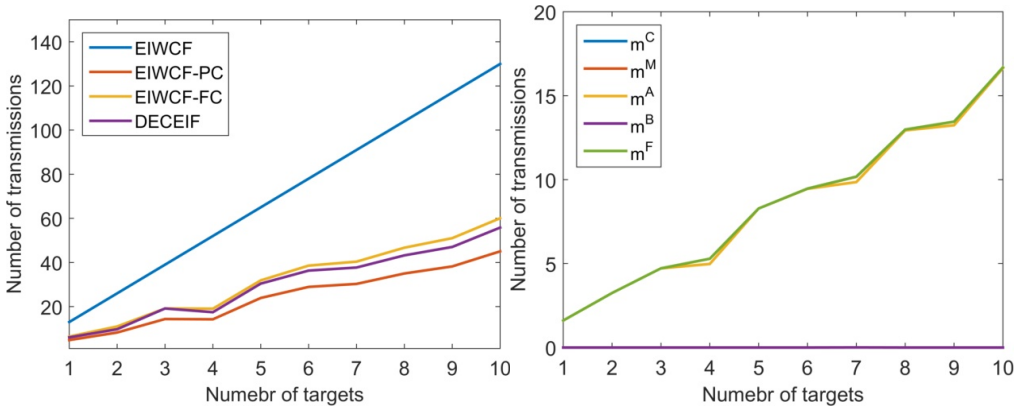


Figure 3.8: Communication cost with the increasing numbers of targets in a Type II network. The vertical axis indicates the number of transmissions during coalition formation and tracking fusion, averaged by the number of cameras and experimental time steps. (a) Communication cost of distributed and decentralised methods. (b) Transmission break-down of each message with EIWCF-FC.

3.4 Summary

This Chapter focused on the coalition formation among static cameras prior to target tracking. We proposed a communication-aware coalition formation scheme prior to distributed tracking to address limited network bandwidth. With simulations, we demonstrated that the proposed coalition formation supports a larger number of concurrent tracking tasks with a quicker convergence compared to distributed tracking without coalition formation. With the modelling of packet loss and packet errors, we demonstrated that the proposed method achieves comparable tracking accuracy and communication cost compared to a decentralised scheme, but enables all cameras within a coalition with the fused target state estimate.

Chapter 4

Agent assignment and motion control for active visual tracking

4.1 Overview

This Chapter focuses on autonomous active visual tracking among camera-equipped agents. Agents move on hearing requests from nearby static cameras and participate in the static-camera tracking coalitions in their proximity to obtain the states of nearby targets, i.e. positions and velocities. Each agent performs local target selection with the objective of maximising the prioritised on-target observation time and computes the robotic control with view maintenance and energy efficiency [C2]. Collision avoidance among agents and targets is not accounted in this Chapter but is detailed in Chapter 5.

This Chapter is organised as follows. Section 4.2 describes the proposed local decision-making and motion control for multi-agent active visual tracking; Subsection 4.2.1 introduces the proposed criterion for agents to locally select their target to track. Subsection 4.2.2 introduces the energy-efficient controller for active visual tracking. Section 4.3 presents the method evaluation with simulations; Subsection 4.3.1 evaluates the controller performance in terms of view maintenance and energy efficiency using different trajectories. Subsection 4.3.2 evaluates the strategy of agents for local target selection. Finally, Section 4.4 summarises this chapter.

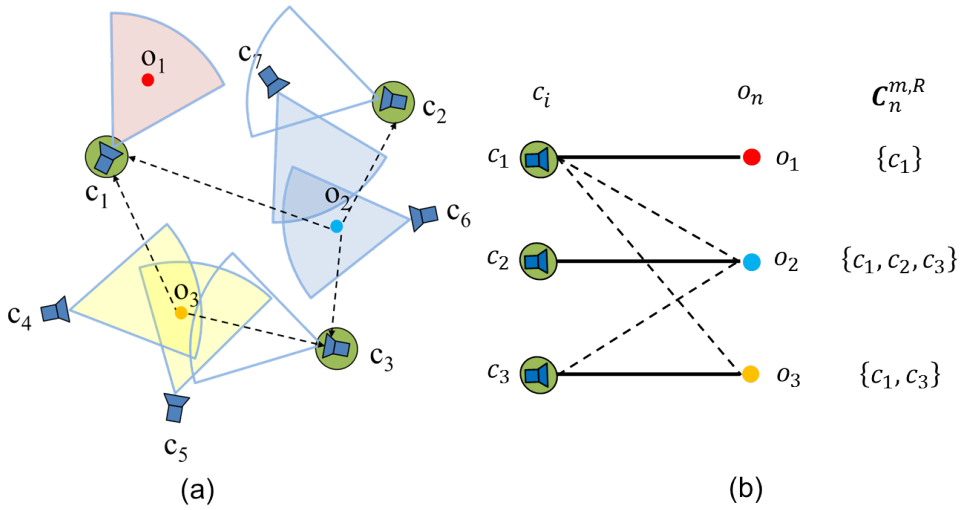


Figure 4.1: Demonstration for agent local target selection. (a) Example scenario with cases where multiple agents receive the request to track same target, e.g. agent c_1, c_2 and c_3 receive the request for target o_2, and agents need to select among multiple candidate targets, e.g. agent c_1 needs to select among o_1, o_2 and o_3. (b) The graph representation of (a), with the solid line indicates the results of the local target selection.

4.2 Proposed method

Static cameras continuously perform distributed tracking on targets and identify those that need to be actively tracked by agents with tracking priorities. Agents move on demand and perform target selection when hearing requests from static cameras. Let $\Lambda_i(t)$ be the set of candidate targets that agent c_i receives at t. Each agent first independently selects a target among $\Lambda_i(t)$ and then computes its robotic control for actively visual tracking once a target is selected.

4.2.1 Agent target selection

If target o_n with tracking priority w_n is not tracked by an agent, the static cameras that are tracking o_n will send requests to their nearby agents. To avoid redundant transmission of request messages, only the static camera that is closest to an agent sends a request message for tracking o_n, $m_i^{n,R}$ (superscript R for 'Request'). $m_i^{n,R}$ contains the estimated target state $\mathbf{s}_i^n$, the tracking priority w_n and the states of other agents that receive the same request. These information is used by an agent to locally select its target.

Let $\mathbf{C}_n^{m,R}$ be the set of agents that receive the request for tracking o_n. Each agent independently selects a target among Λ_i based on a criterion that aims to maximise the prioritised observation on targets while reducing the risk of one target being selected by multiple agents. To

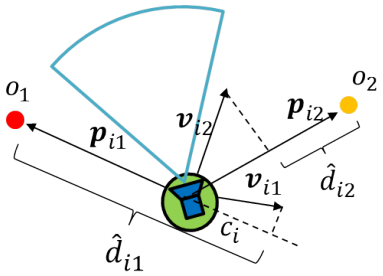


Figure 4.2: Example of the distance-based cost $\hat{d}_{in}$ for an agent c_i to capture two candidate targets o_1 and o_2. $\mathbf{p}_{in}$ and $\mathbf{v}_{in}$ are relative position and velocity between c_i and o_n, respectively.

allow an agent flexibly switch to track a target with a higher tracking priority, we treat the target that c_i is currently tracking as a candidate target during target selection. See example scenario in Fig. 4.1.

For each $o_n \in \Lambda_i$, agent c_i estimates a utility α_i^n that relates to the target priority w_n and the cost for the agent to capture the target within the FoV:

$$\alpha_i^n = w_n g_i^n, \tag{4.1}$$

where g_i^n is a cost-efficiency utility that is inversely proportional to the energy cost for an agent to capture the target. Energy cost is often approximated by the distance between an agent and a target [98]. In addition to distance, we take into account the relative velocity between an agent and a target to achieve a more accurate approximation. We construct the distance-based cost as:

$$\hat{d}_{in} = \left\| \mathbf{p}_{in} - \frac{\mathbf{p}_{in} \cdot \mathbf{v}_{in} \Delta T}{\|\mathbf{p}_{in}\|} \right\|, \tag{4.2}$$

where $\mathbf{p}_{in}$ and $\mathbf{v}_{in}$ are the relative position and velocity, respectively. $\frac{\mathbf{p}_{in} \cdot \mathbf{v}_{in} \Delta T}{\|\mathbf{p}_{in}\|}$ is the projection of the agent's displacement in ΔT time assuming constant relative velocity $\mathbf{v}_{in}$ onto the relative position $\mathbf{p}_{in}$. We choose $\Delta T = 1$ s for simplicity. Each agent obtains the states of candidate targets from received request messages. One example of the distance-based cost in shown in Fig. 4.2.

In order to combine g_i^n with the tracking priority of the target at a similar scale, we first divide $\hat{d}_{in}$ by the maximum request range and then compute the g_i^n as:

$$g_i^n = 1 - \frac{\hat{d}_{in}(t)}{\hat{d}_{\max}}, \tag{4.3}$$

where $\hat{d}_{\max}$ is the maximum request range which is set to $r_v + r_c$.

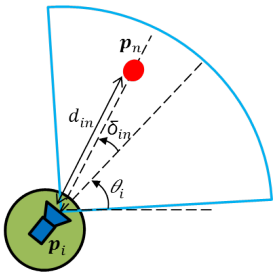


Figure 4.3: Top-view of an agent c_i that performs active visual tracking on a target o_n. c_i locates at $\mathbf{p}_i$ facing θ_i and o_n locates at $\mathbf{p}_n$. d_{in} is the distance from the agent to the target and δ_{in} is the deviation angle from the agent heading direction to its target.

Agents that receive the same request may not be able to communicate with each other directly, which can result in multiple agents selecting the same target to track. Agents should discount the utility α_i^n for the target with more agents receiving the same request. Let $\alpha_i^{n,l}$ be the utility for local selection:

$$\alpha_i^{n,l} = \alpha_i^n \frac{g_i^n}{\sum_{c_j \in \mathbf{C}_n^{m,R}} g_j^n}. \tag{4.4}$$

Agents select the target with the highest $\alpha_i^{n,l}$ to track. However the criterion cannot avoid multiple agents tracking the same target. Since agents join in distributed tracking with static cameras, once agents know if there is the presence of other agents tracking the same target, the agent with the highest information utility continues tracking while the other agents switch to idle.

4.2.2 Motion control for active visual tracking

An agent computes its robotic control vector $\mathbf{u}_i$ once a target is selected, with the objective of maintaining its target centred at the FoV while minimising the energy cost. The control vector is computed by minimising the weighted sum of two cost functions:

$$\mathbf{u}_i = \text{argmin}_{\mathbf{u}} \left(\lambda J_1 + (1-\lambda) J_2\right), \tag{4.5}$$

where J_1 is the viewing cost that penalises the deviation of the target position from the FoV centre. J_2 the energy cost that penalises the accelerations/decelerations of the agent's motion. λ is the weight assigned to the viewing cost J_1. In the case of two objectives, we use the exponential function to emphasise the penalty on the deviation of each desired criterion [70].

Viewing cost J_1 is computed from the distance between the agent and its target, d_{in}, and the

deviation angle from the agent heading to its target, δ_{in} (see Fig. 4.3). Let $\rho_{in}^{d} = \frac{2\left|d_{in} - \frac{1}{2}r_v\right|}{r_v}$ and $\rho_{in}^{\delta} = \frac{2\delta_{in}}{\phi}$ be the ratio of the distance and deviation angle to the FoV centre with respect to the half view range $\frac{r_v}{2}$ and half view angle $\frac{\phi}{2}$. We estimate d_{in} and δ_{in} based on the predicted target state and the resulted state of c_i given the control vector $\mathbf{u}_i$. J_1 is computed as:

$$J_1 = \exp\left(\sqrt{{\rho_{in}^{d}}^2 + {\rho_{in}^{\delta}}^2}\right). \tag{4.6}$$

$J_1 < \exp(1)$ ensures that the target locates within the camera's FoV. $\sqrt{{\rho_{in}^{d}}^2 + {\rho_{in}^{\delta}}^2}$ is the view deviation ratio. The minimum, $J_1 = 1$, is achieved when the target locates at the centre of the FoV.

Energy cost J_2 is computed from the accelerations/decelerations of the agent because accelerations consume more energy and decelerations lead to heat loss. Let $\Delta\mathbf{v}_i$ be the velocity difference between the resulted velocity of c_i given the control $\mathbf{u}_i$ and the current velocity $\mathbf{v}_i$. We compute J_2 as:

$$J_2 = \exp\left(\frac{\|\Delta\mathbf{v}_i\|}{v_{\max} + \|\mathbf{v}_i\|}\right), \tag{4.7}$$

where $v_{\max} + \|\mathbf{v}_i(t)\|$ is the maximum speed difference of an agent between two consecutive time steps. The minimum, $J_2 = 1$, is achieved when the agent does not change velocity. The cost increases monotonically up to $\exp(1)$ as the difference of velocity increases.

The weight λ lies in range of $[0.5\ ,\ 1]$. $\lambda \geq 0.5$ because the main objective is to maintain the target centred at the camera's FoV. The solution of the minimisation lies within a space spanned by the translational speed v and the rotational speed ω. We solve the problem via brute-force searching in a discrete space, as the complexity is low and a global optimum can be guaranteed. When the agent loses its target out of the FoV, the objective is to re-capture the target inside the camera's FoV as soon as possible, we therefore set $\lambda = 1$ without accounting for energy efficiency. When the agent has its target inside the FoV, the objective is to maintain the target centred at the FoV in an energy-efficient manner. The weight λ can be chosen experimentally for the best trade-off between the decrement of J_1 and the increment of J_2. We explain in details the weight tuning procedure and the justification of the cost options in Subsection 4.3.1.

4.3 Validation

In this section, we firstly investigate the weighting parameter in the proposed controller in order to optimally set the weight, then justify the choices of the exponential cost with the comparison to the quadratic cost, and we finally validate the improvement in terms of energy efficiency and view maintenance with comparison to an optimal controller that only accounts for view performance and a stable feedback controller. We then validate the agent target selection strategy in terms of the prioritised observation time on targets and energy efficiency, and compare it with a local selection strategy only using distance-based criterion and a global agent-target assignment strategy.

4.3.1 Evaluation of motion controller

The proposed controller minimises a weighted sum of two cost functions whose weight setting influences the controller behaviours. With synthetic trajectories and trajectories that are extracted from people, we investigate the trade-off between the two costs at different λ, i.e. the weight assigned to view cost J_1, in order to find the most cost-efficient setting.

We define the cost efficiency, η_λ, as the ratio of the cost reduction of average view cost with $\lambda \in [0.6,\ 1]$ compared to the one with $\lambda = 0.5$ to the increment of average energy cost with $\lambda \in [0.6,\ 1]$ compared to the one with $\lambda = 0.5$:

$$\eta_\lambda = \frac{\left|\bar{J}_{1,\lambda} - \bar{J}_{1,0.5}\right|}{\left|\bar{J}_{2,\lambda} - \bar{J}_{2,0.5}\right|}, \tag{4.8}$$

where $\bar{J}_{1,\lambda}$ ($\bar{J}_{2,\lambda}$) is the view cost (energy cost) averaged by the experiment time and the number of trajectories with a certain λ.

With the most cost-efficient λ, we then evaluate the view performance and energy cost with the proposed controller, and compare it with controllers that only considers view performance, i.e. one optimal controller [137] and one feedback controller [73].

Experiment setup

The synthetic trajectories consist of three sets of trajectories with variations in either direction or speed by adding noises to a speed at 1 m/s. The first set of trajectories (Set I) has targets moving at constant speed but with time-varying direction that is achieved by adding bounded Gaussian noises to the current direction. The second set of trajectories (Set II) has targets moving at a

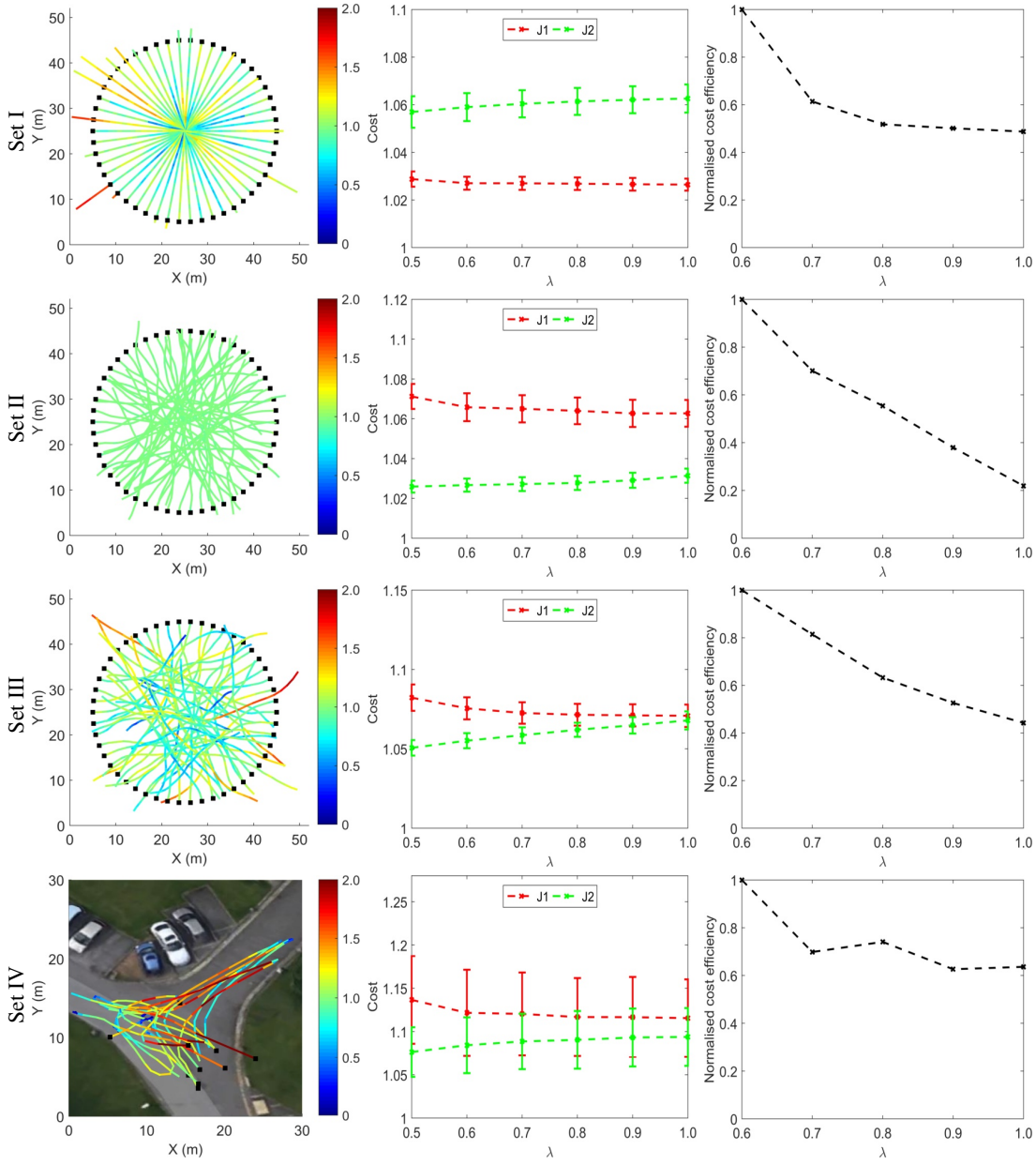
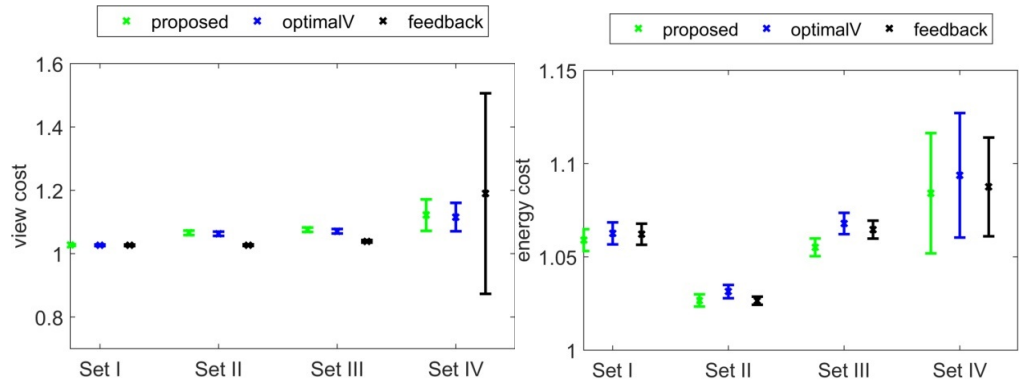


Figure 4.4: Four sets of trajectories and their corresponding cost responses and efficiency with varying λ. Left column are the trajectories with black squares indicating the starting positions of targets and the colour along the each trajectory indicating the speed. Middle column shows the cost responses of J_1 and J_2 with increasing λ that are averaged over the number of targets and time steps. Right column shows the cost efficiency (defined in Eq. 4.8) that are normalised by the maximum value among the tested λ values.

fixed direction with the time-varying speed that is achieved by adding bounded Gaussian noises to the constant speed. The third set of trajectories (Set III) has targets moving with both time-varying direction and time-varying speed. Each set consists of 50 trajectories that last 40 time steps. Since the initial position of each trajectory does not affect the controller behaviours, we align the starting position of each trajectory in circle for the ease of visualisation. The final set of



(a) (b)

Figure 4.5: Cost comparison between the proposed controller and an optimal controller (optimalV) and a feedback controller (feedback) that only consider view maintenance. (a) Results of view cost (J_1). (b) Results of energy cost (J_2).

trajectories (Set IV) consists of 10 real people trajectories with 60 s duration extracted from the PETS2009 S2L1 sequence with provided camera calibration[1]. The maximum target speed in all four sets of trajectories is 2 m/s. The left column of Fig. 4.4 shows the ground-plane trajectories with the colour indicating the speed of a target.

The robotic platform follows a differential-drive kinematic model with the maximum speed of 3 m/s (larger than the target speed to ensure successful capture) and maximum angular speed of π rad/s. The agent is initialised with the target centred at its FoV. Agents update their controls every second and executes the control at the frequency of 10 Hz. The camera equipped on the robotic platform has a sector-shaped FoV to simulate the side view. The FoV has a view angle of 90° and view range of 5 m.

Results discussion

Fig. 4.4 shows the cost responses and the cost efficiency (normalised by the maximum cost efficiency value) with different λ, where the results of cost responses are in the middle column and the cost efficiency are in the right column, respectively. The results are averaged over the number of trajectories in each set and the time steps. In general the cost efficiency with four sets of trajectories follows a consistent decreasing trend as λ increases, and $\lambda = 0.6$ gives the best cost efficiency. In addition, we find that trajectories with various speeds (Set I) lead to more energy cost compared to the one resulted from targets with various directions (Set II).

Fig. 4.5 shows the comparison between the proposed controller, the optimal controller and

[1]http://www.cvg.reading.ac.uk/PETS2009. Last accessed: 07/08/2017

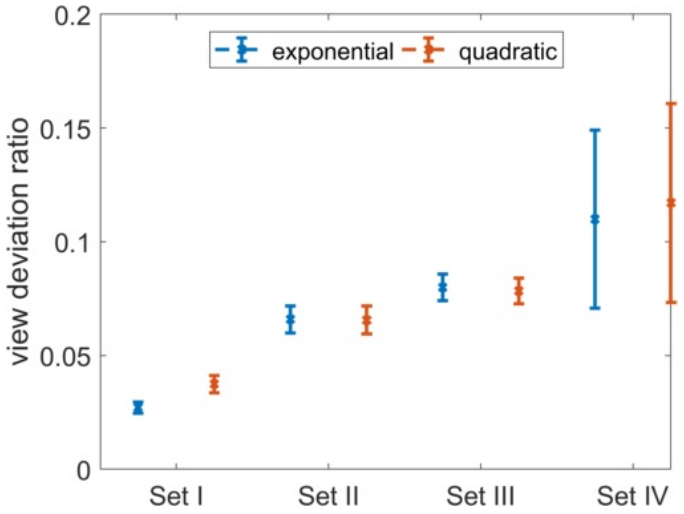


Figure 4.6: View deviation ratio with the quadratic cost and the exponential cost in the case of $\lambda = 0.5$ using four sets of trajectories.

the feedback controller with the four sets of trajectories. Fig. 4.5(a) and (b) show the view cost and energy cost, respectively, averaged over time and the number of targets. We can see the proposed controller has slightly higher view cost compared to the optimal controller with the gain in reduced energy cost with all four sets of trajectories. The feedback controller can achieve equivalent energy cost compared to the proposed controller with slightly lower view cost with the synthetic trajectories, however its view cost with the real people trajectories is higher with a large variance. The feedback controller computes the control of an agent only based on the error between its current position to its desired position. When the tracked target has sudden velocity changes as in Set IV, the feedback controller may not be able to react promptly and therefore lead to a worse view performance.

Fig. 4.6 shows the view deviation ratio (defined in Eq. 4.6) with the quadratic cost and exponential cost using four sets of trajectories. The weight of each cost function is set equally, i.e. $\lambda = 0.5$. With the account of energy cost function, the view cost in the exponential form penalises more when the deviation to the desired view is large, and therefore keeps the view deviation smaller compared to that obtained by the quadratic cost function.

4.3.2 Evaluation of local target selection

We finally evaluate the proposed local target selection criterion performed independently by each agent (Dis-U) in terms of on-target prioritised observation time and energy consumption, and compare it with (i) a centralised assignment using Hungarian algorithm (Cen-U) and (ii) a local

target selection strategy that only accounts for the distance cost [98] (Dis-D). The centralised assignment serves as a performance upper bound using knowledge of all agents and targets. The centralised strategy updates the assignment at each time step by computing the generic utility for each agent selecting each candidate target, i.e. α_i^n.

The observation performance is quantified by the *prioritised observation ratio*, which is the ratio of the duration when a target is observed by its agent weighted by the tracking priority of the target:

$$\frac{1}{TW}\sum_{t=1}^{T}\sum_{n=1}^{N} b_n(t)w_n, \tag{4.9}$$

where N the number of targets, $b_n(t)$ is a binary value indicating whether o_n is observed by its agent at t, $W = \sum_{n=1}^{N} w_n$ and T is the experimental time. Targets with higher priority being observed longer will lead to a larger prioritised observation ratio.

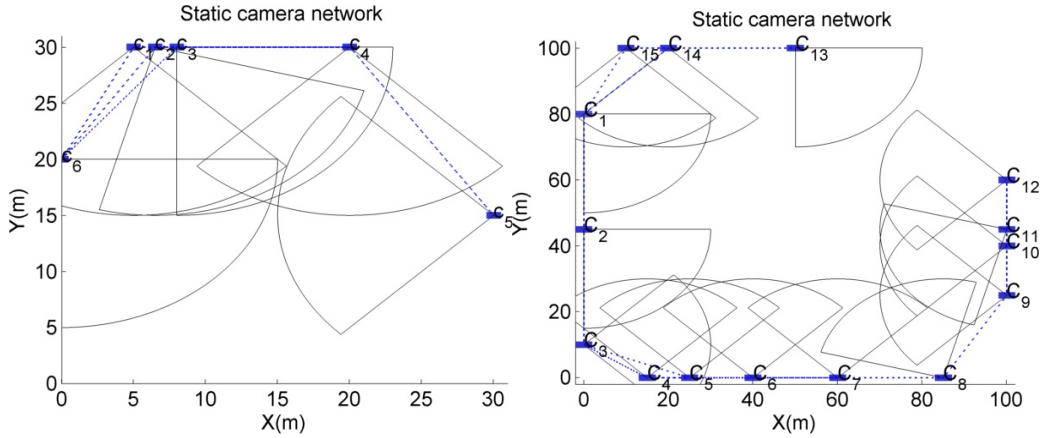
The energy consumption is computed as [78] and takes into account both kinetic energy and the energy to overcome surface friction. The two energy components are normalised by their maximum consumption independently in order to remove the impacts of physical properties on the results, such as the mass of the platform and the coefficient of friction. Let $E_i(t)$ be the normalised energy consumed by agent c_i at t, we quantify the energy consumption as the *normalised energy cost* which is the normalised energy consumption averaged by the number of targets and time:

$$\frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{|\mathbf{C}^m|} E_i(t), \tag{4.10}$$

where N is the number of targets, T is the experimental time and $|\mathbf{C}^m|$ is the number of agents.

Experiment setup

Two simulated scenarios are tested with static cameras that are placed according to the real deployment in a campus square and a public square. Scenario I is a 30 m$\times$30 m campus square with six static cameras deployed to cover the entrance of shops or buildings (Fig. 4.7(a)), and Scenario II is a 100 m$\times$100 m public square with 15 static cameras deployed to cover the traffics and surroundings of buildings (Fig. 4.7(b)). For both scenarios, the view angle of cameras is set as $\theta = 0.5\pi$ rad. We set the view range $r_v = 15$ m and the communication range $r_c = 20$ m for scenario I, and $r_v = 30$ m and $r_c = 40$ m for scenario II.



(a) (b)

Figure 4.7: Two tested scenarios. Dashed lines indicate whether cameras can communicate (one hop) with each other. (a) Scenario I with six static cameras in a campus square. (b) Scenario II with 15 static cameras in a public square.

Agents are homogeneous, i.e. with the identical shape, sensor modelling and kinematic model, and move freely in the square without the consideration of collision avoidance among other agents and targets. Each agent follows a car-like model with the velocity constraint as $\mathbf{v}_{\max} = [3 \text{ m/s}, \pi \text{ rad/s}]$. The kinematic model of each target is $\mathbf{x} = f(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})$ with $\ddot{\mathbf{x}}$ following a zero-mean bivariate Gaussian distribution of the covariance matrix $diag([0.3\ 0.3])$. The maximum target speed is set to 2 m/s, which is slower than agents in order to guarantee the capture. We initialise the target tracking priority randomly following a uniform distribution and keep the priority constant throughout the experiments. Agents obtain target states by joining in the tracking coalitions with their neighbouring static cameras assuming lossless links without delay, and move with the proposed controller with the cost-efficient setting, i.e. $\lambda = 0.6$. We initialise the location of agents using a uniform distribution. Results shown below are averaged over 50 independent runs.

Results discussion

We test the performance with the increasing numbers of agents under two scenarios for actively tracking five targets with different priorities. Fig. 4.8 (a, b) show the prioritised observation ratio under scenario I and scenario II, respectively. The prioritised observation ratio achieved by the proposed local selection (Dis-U) saturates when the number of agents equals to the number of targets under scenario I, and saturation under scenario II occurs at more agents, i.e. eight agents. This is because in a smaller area (Scenario I) each target is more likely to be tracked by an agent

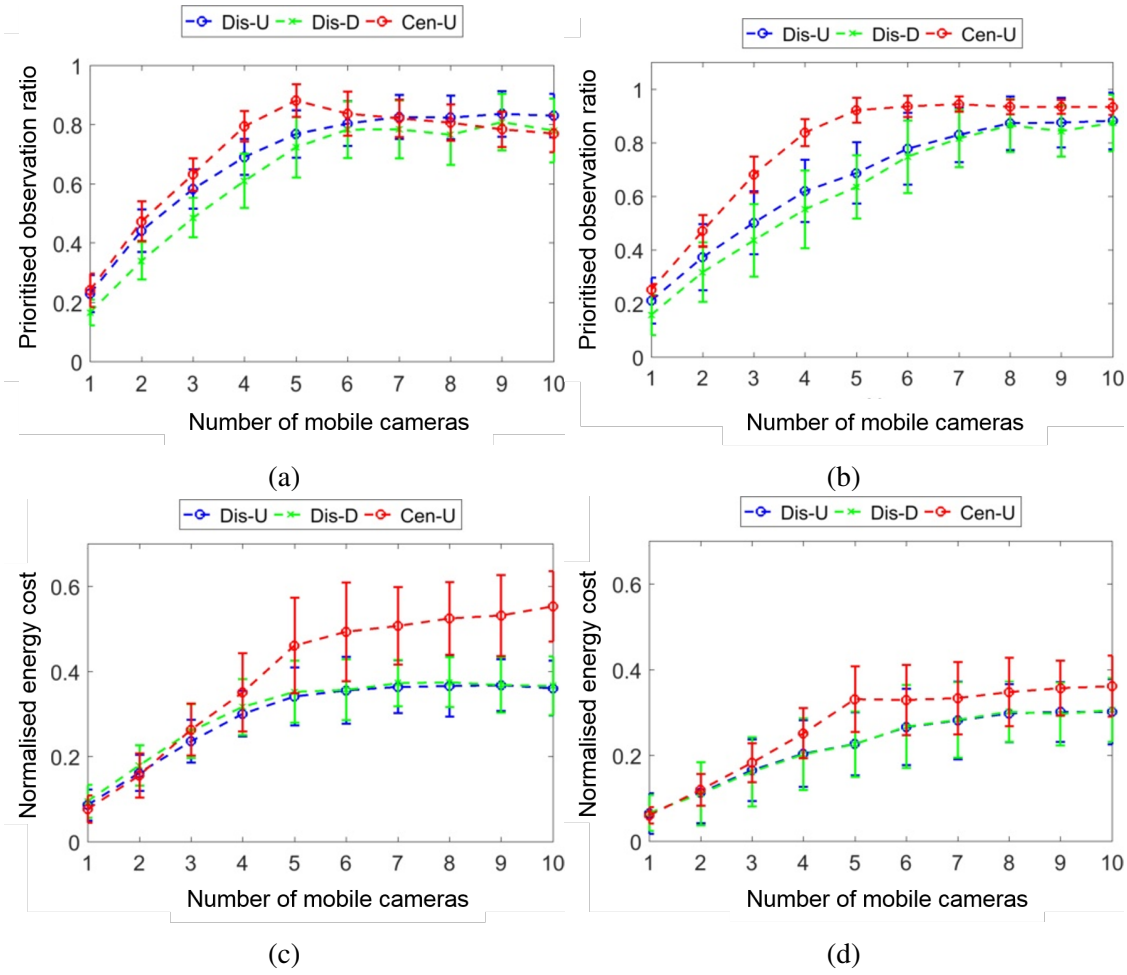


Figure 4.8: The prioritised observation ratio and normalised energy cost with increasing numbers of agents, achieved with the proposed local selection utility (Dis-U), the centralised assignment (Cen-U) and the distance-based selection (Dis-D). (a, b) The prioritised observation ratio under scenario I and scenario II, respectively. (c, d) The normalised energy cost under scenario I and scenario II, respectively.

with neighbourhood communication, while in a larger area (Scenario II) more agents are required due to the limited communication range.

The prioritised observation ratio achieved with Dis-U can approach, but not exceed that with the centralised assignment (Cen-U) when there are less agents than targets. The prioritised observation ratio achieved with Cen-U saturates when there are the same number of agents and targets under both scenarios. However we notice that the prioritised observation ratio achieved by Cen-U may deteriorate as the number of agents increases in a smaller scene (see Fig. 4.8(a)). This is because Cen-U updates the assignment at each time step and results in switching between agents for tracking one target when there are redundant agents. The switching may cause temporary target loss and further lead to reduced prioritised observation ratio and increased energy

cost (Fig. 4.8 (c)). The proposed Dis-U achieves a higher prioritised observation ratio, compared to only distance-based selection (Dis-D) under both scenarios with almost the same energy consumption. Dis-D accounts only distance, which makes the agent-target assignment sensitive to scene dynamics, while the proposed Dis-U improves the prioritised on-target observation by accounting for the relative agent-target velocity and tracking priority.

4.4 Summary

This Chapter focused on agent local decision making on target selection and motion control for energy-efficient active visual tracking. Agents move on request with a local target selection strategy using limited scene knowledge. The prioritised on-target observation time approaches the centralised assignment with increasing numbers of agents without using a global scene knowledge and achieves a higher prioritised observation time compared to the distance-based assignment without consuming more energy. With the real people trajectories, the proposed energy-efficient motion controller reduces the energy consumption by 10% with a little compensation (3%) in the view maintenance, compared to the controller that only accounts for view maintenance.

Chapter 5

Multi-agent active visual tracking with collision avoidance

5.1 Overview

Chapter 3 introduced the coalition formation among static cameras for tracking targets and the strategy for agents to independently select and navigate towards their target without the consideration of collision avoidance [C4]. This Chapter introduces how an agent avoids nearby agents and targets during active visual tracking. This Chapter assumes that static camera coalition formation and agent-target assignment is done, i.e. static cameras know their coalitions and agents know which target to actively track at each time step.

The proposed method is based on the ORCA method and further addresses view maintenance during collision avoidance manoeuvres. We approach view maintenance from two aspects, i.e. an adaptive pair-wise responsibility sharing algorithm and a heading-aware robotic control mapping algorithm. Original ORCA makes each agent to share equal responsibility with another agent when avoiding each other. We adapt this pair-wise responsibility such that the agent with a higher chance of losing its target out of the camera's FoV can share less responsibility for collision avoidance. ORCA computes a collision-free velocity that needs to be further mapped to a feasible robotic control according to the agent's kinematics. Existing control mapping algorithms only aim to achieve the collision-free velocity at each time without accounting for the agent heading direction [127, 128, 5]. This can cause unnecessary target loss when an agent needs to move backward due to either collision avoidance manoeuvres or target moving back-

wards. The heading-aware robotic control mapping algorithm is therefore proposed to minimise the deviation angle from the agent heading to its target in a smooth manner.

This Chapter is organised as follows. Subsection 5.2.1 introduces the ORCA method, followed by the adaptive pair-wise responsibility sharing algorithm in Subsection 5.2.2 and the heading-aware robotic control mapping algorithm in Subsection 5.2.3. Section 5.3 validates the proposed method via simulations using both real people trajectories extracted from publicly available datasets and simulated trajectories using ORCA. Finally conclusion is drawn in Section 5.4.

5.2 Proposed method

At each time step, each agent is aware of the states of itself, nearby targets and agents after jointly tracking with nearby static-camera coalitions. Each agent first computes its preferred velocity without considering the kinematic constraints and exchanges its preferred velocity with neighbouring agents. Each agent then derives the pair-wise velocity constraints induced by each of their neighbouring agents and targets using ORCA with adaptive responsibility sharing. The new collision-free velocity should satisfy all derived velocity constraints and is the closest to the preferred velocity of each agent. The agent finally maps the new collision-free velocity to a feasible robotic control using the proposed heading-aware robotic control mapping algorithm. In this thesis, targets move as they wish without taking actions to avoid agents[1]. Therefore agents avoid targets as passively moving obstacles. We omit t from here to simplify the notation.

5.2.1 Computation of reciprocal collision-avoiding velocities

With the positions and velocities of agent itself and its target, each agent first computes the preferred velocity, i.e. the velocity that the agent would move at as if there were no obstacles in its way. Let $\mathbf{v}_i^*$ be the preferred velocity of agent c_i.

Given that c_i locates at $\mathbf{p}_i$ and its target o_n locates at $\mathbf{p}_n$ moving at velocity $\mathbf{v}_n$, we can compute $\mathbf{v}_i^*$ using a proportional controller:

$$\mathbf{v}_i^* = \mathbf{e}_{in} \max\left(\min\left(K_P\left(d_{in} - d_{in}^*\right),\ v_{max}\right), -v_{max}\right), \tag{5.1}$$

where d_{in}^* is the desired agent-target distance and $K_P > 0$ is the coefficient of the proportional

[1]While this thesis assumes that targets do not react to agents, targets (people) may react to agents in reality for social comforts. Further reading refers to [69]

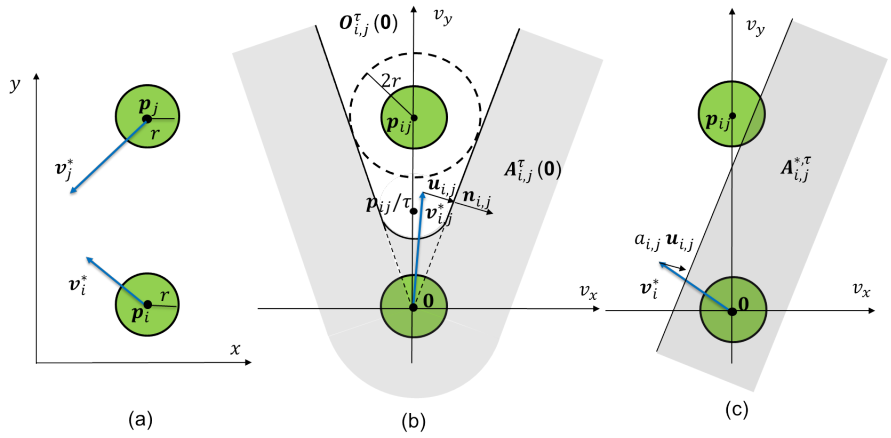


Figure 5.1: Optimal reciprocal collision-avoiding velocities. (a) Agent c_i and c_j with radius r at $\mathbf{p}_i$ and $\mathbf{p}_j$ with their preferred velocity $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ (indicated by blue arrows), respectively. (b) Grey shadow area ($\mathbf{A}_{i,j}^{\tau}(\mathbf{0})$) indicates the relative velocities of c_i that are collision-avoiding to c_j in τ time steps. $\mathbf{u}_{i,j}$ is the minimal velocity for the relative velocity of c_i to c_j, $\mathbf{v}_{i,j}^*$, to get out of the velocity obstacle ($\mathbf{O}_{i,j}^{\tau}(\mathbf{0})$) and $\mathbf{n}_{i,j}$ is the outward normal at $\mathbf{v}_{i,j}^* + \mathbf{u}_{i,j}$. (c) Grey shadow area indicates the velocities of c_i that are optimal reciprocal collision-avoiding to c_j in τ time steps ($\mathbf{A}_{i,j}^{*,\tau}$ of c_i) when c_i shares $a_{i,j}$ responsibility to avoid c_j.

term. d_{in} is the distance between the agent position $\mathbf{p}_i$ and the predicted target position $\tilde{\mathbf{p}}_n$ in ΔT time which is estimated using the current target velocity, i.e. $\tilde{\mathbf{p}}_n = \mathbf{p}_n + \mathbf{v}_n \Delta T$. We set $\Delta T = 1$ s for simplicity and the value of K_P is set to 1 in experiments for smooth motion. $\mathbf{e}_{in}$ is a unit vector that indicates the direction from $\mathbf{p}_i$ towards $\tilde{\mathbf{p}}_n$. The min-max operation constrains the agent's speed within the speed limit $v_{\max}$.

Original ORCA paradigm assumes that each agent is able to infer the preferred velocities of other agents, which may not be practical when each agent is actively tracking a moving target, i.e. the preferred velocity is time-varying and unpredictable. We allow each agent to explicitly exchange its preferred velocity with neighbouring agents assuming that the range for collision avoidance is smaller than the communication range. As for targets, each agent takes the current velocity of a nearby target as the preferred velocity of that target.

With knowledge of the shapes, positions and velocities of nearby agents and targets, together with their preferred velocities, each agent can derive the pair-wise velocity constraint induced by each nearby agent and target. Let us consider a pair of agents c_i and c_j at position $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively, aiming to achieve their preferred velocity $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$, respectively (Fig. 5.1(a)). In order to derive the set of collision-avoiding velocities of c_i with respect to c_j, we first derive the Velocity Obstacle (VO) induced by c_j, i.e. the set of velocities of agent c_i that can lead to a

collision with c_j in a time horizon τ [43]. The relative velocity space of c_i is considered to derive VO as shown in Fig. 5.1(b).

In the relative velocity space of c_i, the preferred velocity of c_i becomes $\mathbf{v}_{i,j}^* = \mathbf{v}_i^* - \mathbf{v}_j^*$ and the preferred velocity of c_j becomes $\mathbf{0}$. Let $\mathbf{O}_{i,j}^{\tau}(\mathbf{0})$ be the VO of c_i induced by c_j assuming c_j moves at its preferred velocity, i.e. $\mathbf{0}$, in the relative velocity space:

$$\mathbf{O}_{i,j}^{\tau}(\mathbf{0}) = \{\mathbf{v} \mid \|t\mathbf{v}\| \geqslant \left\|\mathbf{p}_{ij} - 2r\right\|, \, t \in [0,\tau]\}, \tag{5.2}$$

where $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ is the relative position of c_j with respect to c_i.

The set of collision-avoiding relative velocities for c_i to avoid c_j in τ time horizon, $\mathbf{A}_{i,j}^{\tau}(\mathbf{0})$, can be therefore represented as:

$$\mathbf{A}_{i,j}^{\tau}(\mathbf{0}) = \{\mathbf{v} \mid \mathbf{v} \notin \mathbf{O}_{i,j}^{\tau}(\mathbf{0})\}. \tag{5.3}$$

Reciprocal collision avoidance occurs when c_i and c_j choose to move at $\mathbf{v}_i \in \mathbf{A}_{i,j}^{\tau}(\mathbf{v}_j)$ and $\mathbf{v}_j \in \mathbf{A}_{j,i}^{\tau}(\mathbf{v}_i)$, respectively [12]. ORCA defines the set of velocities that is not only reciprocal collision-avoiding but also the closest to the preferred velocity of each agent.

When $\mathbf{v}_{i,j}^*$ lies within the VO as shown in Fig. 5.1(b), ORCA aims to shift $\mathbf{v}_{i,j}^*$ out of the VO with a minimal effort that is contributed by both agents. Let $\mathbf{u}_{i,j}$ be the vector starting from $\mathbf{v}_{i,j}^*$ to the closest point at the boundary of the VO (see Fig. 5.1(b)). $\mathbf{u}_{i,j}$ is the minimal relative velocity changes between c_i and c_j to avoid collisions within τ. $\mathbf{n}_{i,j}$ is the outward plane normal at $\mathbf{v}_{i,j}^* + \mathbf{u}_{i,j}$.

Each agent shares a partial responsibility for collision avoidance. Let $a_{i,j}$ and $a_{j,i}$ be the responsibility c_i and c_j take to avoid each other, respectively, and $a_{i,j} + a_{j,i} = 1$. The responsibility $a_{i,j}$ indicates that how much c_i will compensate $\mathbf{u}_{i,j}$ in order to shift $\mathbf{v}_{i,j}^*$ out of the VO.

The set of optimal reciprocal collision-avoiding velocity for c_i to avoid c_j in τ time steps is defined as:

$$\mathbf{A}_{i,j}^{*,\tau} = \{\mathbf{v} \mid \mathbf{v} - (\mathbf{v}_i^* + a_{i,j}\mathbf{u}_{i,j}) \cdot \mathbf{n}_{i,j} \leqslant 0\}. \tag{5.4}$$

$\mathbf{A}_{i,j}^{*,\tau}$ are the velocities that lie in the half-plane (the plane in grey shadow in Fig. 5.1(c)) in the direction of $\mathbf{n}_{i,j}$, after shifting the VO by $\mathbf{v}_i^* + a_{i,j}\mathbf{u}_{i,j}$. In the same way, we can construct the set of collision-avoiding velocities of c_j induced by c_i, i.e. $\mathbf{A}_{j,i}^{*,\tau}$.

Each agent avoids targets as passively moving obstacles in this thesis by setting the responsibility for avoiding a target to 1. Let $\mathbf{A}_{i,n}^{*,\tau}$ be the set of velocities of c_i that are collision-avoiding to a target o_n in τ time steps. Each agent c_i estimates the pair-wise $\mathbf{A}_{i,j}^{*,\tau},\ \forall c_j \in \mathbf{C}_i^A$, and $\mathbf{A}_{i,n}^{*,\tau},\ \forall o_n \in \Lambda_i^A$, where $\mathbf{C}_i^A$ refers to the set of agents within the avoidance range of c_i and similarly, Λ_i^A is the set of targets that are within the collision avoidance range (superscript A is short for Avoidance).

Let $\mathbf{V}_i$ be the set of velocities that are accessible under the speed/acceleration limits. The final set of accessible velocities of c_i that are collision-avoiding to all agents and targets in its collision avoidance range, $\mathbf{A}_i^{*,\tau}$, is:

$$\mathbf{A}_i^{*,\tau} = \left(\bigcap_{c_j \in \mathbf{C}_i^A} \mathbf{A}_{i,j}^{*,\tau} \right) \cap \left(\bigcap_{o_n \in \Lambda_i^A} \mathbf{A}_{i,n}^{*,\tau} \right) \cap \mathbf{V}_i. \tag{5.5}$$

Note that the collision avoidance manoeuvre for the target that an agent is tracking, can deteriorate the viewing performance of the agent. We therefore exclude the target being tracked by agent c_i from Λ_i^A unless this target is considered to be 'too close' to the agent. We set the collision avoidance range for the target being actively tracked by an agent to half of the preferred distance in this thesis. Each agent c_i finally computes a new collision-avoiding velocity $\mathbf{v}_i^A$ (superscript A is short for Avoidance) that lies within $\mathbf{A}_i^{*,\tau}$ and is the closest to its preferred velocity.

Note that when the neighbouring agents or targets are densely around an agent, the agent may have no accessible collision-avoiding velocity, i.e. $\mathbf{A}_i^{*,\tau} = \emptyset$ (an empty-set case). ORCA handles the empty-set cases by allowing the agent to intrude the half-plane velocity constraints with a small vector until there is one accessible collision-avoiding velocity [12]. The resulted velocities of agents depend on their neighbouring agents rather than their preferred velocity. Collision avoidance is also not guaranteed as the velocity constraint is adjusted only for one agent, instead of for a pair of agents. This can lead to potential collisions when the other agent is close to the agent. [C3] addresses the empty-set cases by preventing the occurrences of the empty sets with the intuitive that an agent with a smaller set of accessible collision-avoiding velocities should take less responsibility when avoiding another agent. This thesis does not focus on addressing the empty-set cases and adopts the strategy in [12].

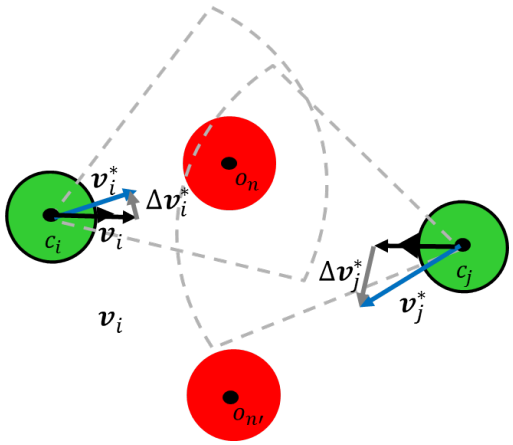


Figure 5.2: Agent c_i is actively tracking o_n and agent c_j is actively tracking $o_{n'}$. For each agent, the black arrow indicates the current velocity, the blue arrow indicates the preferred velocity and the grey arrow indicates the difference between the current velocity and the preferred velocity.

5.2.2 Adaptive pair-wise responsibility sharing

Each agent shares equal responsibility with another agent when deriving the pair-wise velocity constraint, i.e. $a_{i,j} = a_{j,i} = 0.5$, in majority of the ORCA literature [12, 128, 5, 7, 11]. However, the choice of the pair-wise responsibility influences the number and distribution of accessible collision-avoiding velocities between a pair of agents. The pair-wise responsibility has been exploited to assign right of way in crowd simulations [29] and reducing the occurrences of empty-set cases in densely-intersecting multi-agent scenarios [C3]. As agents aim to maintain their target within the FoV of their on-board camera, it is preferable to assign less responsibility to an agent with a higher risk of losing its target, so that the agent can have more chances to move at its preferred velocity, i.e. the velocity that maintains its target centred at the FoV.

When a target moves away from the desired position within its agent's FoV, the preferred velocity of the agent (computed using Eq. 5.1) will differ more from its current velocity (see Fig. 5.2). Let $\Delta\mathbf{v}_i^*$ be the difference between the preferred velocity $\mathbf{v}_i^*$ and the current velocity $\mathbf{v}_i$. We can estimate the risk level, q_i, of c_i losing its target based on $\Delta\mathbf{v}_i^*$:

$$q_i = exp\left(|\Delta\mathbf{v}_i^*|\right). \tag{5.6}$$

The exponential function is applied to $|\Delta\mathbf{v}_i^*|$ in order to maintain $q_i > 0$ and to aggravate the risk level as the velocity difference increases.

The responsibility that c_i shares with c_j, i.e. $a_{i,j}$, depends on how q_i is variant to q_j. The variance is opposite to the concept of fairness which quantifies how alike two values are. We

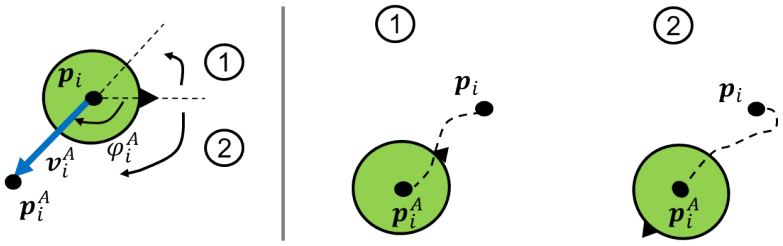


Figure 5.3: An agent locating at $\mathbf{p}_i$ has a collision-avoiding velocity $\mathbf{v}_i^A$ to achieve, where $\mathbf{p}_i^A$ is the resulted temporary goal position to reach in the next second with velocity $\mathbf{v}_i^A$. φ_i^A is the angle from agent current heading direction to position $\mathbf{p}_i^A$. The agent can achieve $\mathbf{v}_i^A$ with two options. Option 1: the agent moves backward with the complement of φ_i^A to the left-hand side; Option 2: the agent moves forward with φ_i^A to the right-hand side.

adopt the Jain's fairness measure [55] to compute $a_{i,j}$ as the measure provides a continuous and bounded value. Let ρ_{ij} be the fairness between q_i and q_j:

$$\rho_{ij} = \frac{(q_i + q_j)^2}{2({q_i}^2 + {q_j}^2)}, \tag{5.7}$$

where $\rho_{ij} \in [0.5,\ 1]$ with 0.5 representing the least fair and 1 representing the most fair. We finally compute $a_{i,j}$ as:

$$a_{i,j} = \begin{cases} \rho_{ij} - 0.5 & q_i > q_j \\ 1.5 - \rho_{ij} & q_i \leqslant q_j \end{cases} \tag{5.8}$$

The responsibility $a_{j,i}$ for c_j to avoid c_i is computed in the same way. The proposed adaptive algorithm guarantees that $a_{i,j} + a_{j,i} = 1$.

5.2.3 Heading-aware robotic control mapping

The collision-avoiding velocity $\mathbf{v}_i^A$ is computed without considering the agent's kinematic constraints that needs to be further mapped to a feasible robotic control. $\mathbf{v}_i^A$ sets a temporary goal position for c_i to reach in the next second, i.e. $\mathbf{p}_i^A = \mathbf{p}_i + \mathbf{v}_i^A$. Feedback-based controllers are commonly used to compute the robotic control vector $\mathbf{u}_i$ to reach the temporary goal position at each time step [127, 128, 6, 5]. The feedback controller takes as input the distance error, $d_{i,e}$, between $\mathbf{p}_i$ and $\mathbf{p}_i^A$ and the angle error, $\varphi_{i,e}$, from current agent heading to $\mathbf{p}_i^A$.

We employ the controller proposed in [73] for its proven stability. Given the distance error

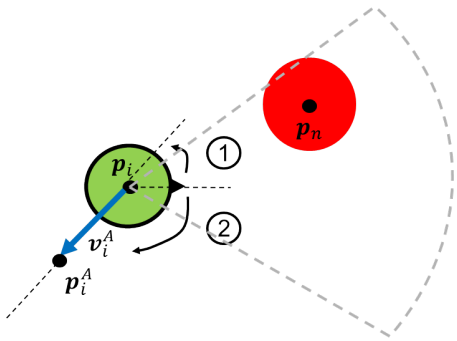


Figure 5.4: Agent c_i locates at $\mathbf{p}_i$ and aims to achieve its collision-free velocity $\mathbf{v}_i^A$. The agent will lose its target out of its FoV if Option 2 is selected whereas Option 1 maintains its target within its FoV.

$d_{i,e}$ and angular error $\varphi_{i,e}$, the control is computed as:

$$\begin{aligned} v_i &= \kappa_1 d_{i,e} \cos(\varphi_{i,e}) \\ \omega_i &= \kappa_1 d_{i,e} \sin(\varphi_{i,e}) \cos(\varphi_{i,e}) + \kappa_2 \varphi_{i,e}, \end{aligned} \tag{5.9}$$

where κ_1 and κ_2 are two parameters for the feedback controller that affect the agility of motion. κ_1 and κ_2 can be tuned experimentally. We set $\kappa_1 = 1$ and $\kappa_2 = 3$ for smooth motion.

The sign of $d_{i,e}$ can be either positive or negative. Positive $d_{i,e}$ leads to a forward movement, while negative $d_{i,e}$ leads to a backward movement. As shown in Fig. 5.3, an agent has two options to reach its temporary goal position. The agent can either move forward while turning φ_i^A to the right side of the agent, or move backward while turning the complement angle of φ_i^A to the left side of the agent. $\varphi_i^A \in (-\pi, \pi]$ is the angle from agent heading direction to $\mathbf{p}_i^A$. Both options achieve the same $\mathbf{v}_i^A$ but may result in different deviation angles from the agent heading direction to its target position.

In navigation applications, the sign of $d_{i,e}$ is set to be positive [6, 127] (the second option shown in Fig. 5.3) and this can cause unnecessary target loss as shown in Fig. 5.4.

We aim to minimise the deviation angle from the agent heading to its target by properly setting the sign of $d_{i,e}$. Let $d_{i,e}^+$ and $\varphi_{i,e}^+$ be the distance error and the angular error of a forward movement, respectively. We set $d_{i,e}^+$ and $\varphi_{i,e}^+$ as:

$$\begin{aligned} d_{i,e}^+ &= \left\| \mathbf{v}_i^A \right\| \\ \varphi_{i,e}^+ &= \varphi_i^A. \end{aligned} \tag{5.10}$$

Correspondingly, $d_{i,e}^{-}$ and $\varphi_{i,e}^{-}$ are the distance and angular error of a backward movement, which are set as:

$$\begin{aligned} d_{i,e}^{-} &= -\left\|\mathbf{v}_i^A\right\| \\ \varphi_{i,e}^{-} &= \begin{cases} \varphi_i^A - \pi, & \varphi_i^A > 0 \\ \varphi_i^A + \pi, & \varphi_i^A <= 0 \end{cases} \end{aligned} . \tag{5.11}$$

We compute the candidate robotic control vectors of the two options using Eq. 5.9. Let $\mathbf{u}_i^{+} = \left[v_i^{+},\ \omega_i^{+}\right]$ be the robotic control vector when the agent has a forward movement, and $\mathbf{u}_i^{-} = \left[v_i^{-},\ \omega_i^{-}\right]$ be the robotic control vector for a backward movement.

Let $\Delta\delta_{in}$ be the difference of the deviation angle from the agent heading to its target between two consecutive time steps [73]:

$$\Delta\delta_{in} = -\omega_i \Delta T + \frac{v_i \Delta T}{d_{in}} \sin(\delta_{in}), \tag{5.12}$$

where ΔT is the time between two consecutive time steps.

Let $\Delta\delta_{in}^{+}$ and $\Delta\delta_{in}^{-}$ be the difference of the deviation angle resulted by the forward movement and backward movement, respectively. The resulted deviation angle at next time step, i.e. in ΔT time, can then be computed as:

$$\delta_{in}(\Delta T) = \delta_{in} + \Delta\delta_{in}. \tag{5.13}$$

Let $\delta_{in}^{+}(\Delta T)$ and $\delta_{in}^{-}(\Delta T)$ be the deviation angle in ΔT time resulted from the forward and backward movement, respectively.

As the objective of active visual tracking is to maintain the target in front of the agent, i.e. $\delta_{in} = 0$, one can simply select the movement direction (forward or backward) that leads to a smaller $|\delta_{in}(\Delta T)|$. However, an agent can encounter oscillations due to continuous sign swapping when the target is positioned orthogonal to the collision-avoiding velocity $\mathbf{v}_i^A$ (see Fig. 5.5). In order to avoid such undesired oscillations and to achieve smooth motion, $|\Delta\delta_{in}|$ can be instead used as the criterion for selecting the movement direction.

However, selecting a smaller $|\Delta\delta_{in}|$ can not guarantee the agent always heading towards its target. We therefore design the robotic control mapping algorithm that combines the two criteria and aims to achieve smooth motion while maintaining the agent heading towards its target.

Algorithm 2 Heading-aware robotic control mapping algorithm that is run on each agent c_i for actively visual tracking its target o_n

Input:

$\mathbf{v}_i^A$: Collision-free velocity of c_i for tracking o_n

δ_{in}: Deviation angle from the heading of c_i to its target o_n

Compute $\left[d_{i,e}^+,\ \varphi_{i,e}^+\right]$ and $\left[d_{i,e}^-,\ \varphi_{i,e}^-\right]$ (Eq. 5.10 and Eq. 5.11)
Compute $\left[v_i^+,\ \omega_i^+\right]$ and $\left[v_i^-,\ \omega_i^-\right]$ (Eq. 5.9)
Compute $\Delta\delta_{in}^+$ and $\Delta\delta_{in}^-$ (Eq. 5.12)
Compute $\delta_{in}^+(\Delta T)$ and $\delta_{in}^-(\Delta T)$ (Eq. 5.13)

if $\left|\Delta\delta_{in}^+\right| \leq \left|\Delta\delta_{in}^-\right|$ **then**
  $\mathbf{u}_i = \mathbf{u}_i^+$
  $\delta_{in}(\Delta T) = \delta_{in}^+(\Delta T)$
else
  $\mathbf{u}_i = \mathbf{u}_i^-$
  $\delta_{in}(\Delta T) = \delta_{in}^-(\Delta T)$
end if

if $|\delta_{in}(\Delta T)| > \frac{\pi}{2}$ **then**
  if $\left|\delta_{in}^+(\Delta T)\right| \leq \left|\delta_{in}^+(\Delta T)\right|$ **then**
    $\mathbf{u}_i = \mathbf{u}_i^+$
  else
    $\mathbf{u}_i = \mathbf{u}_i^-$
  end if
end if

Output: $\mathbf{u}_i$

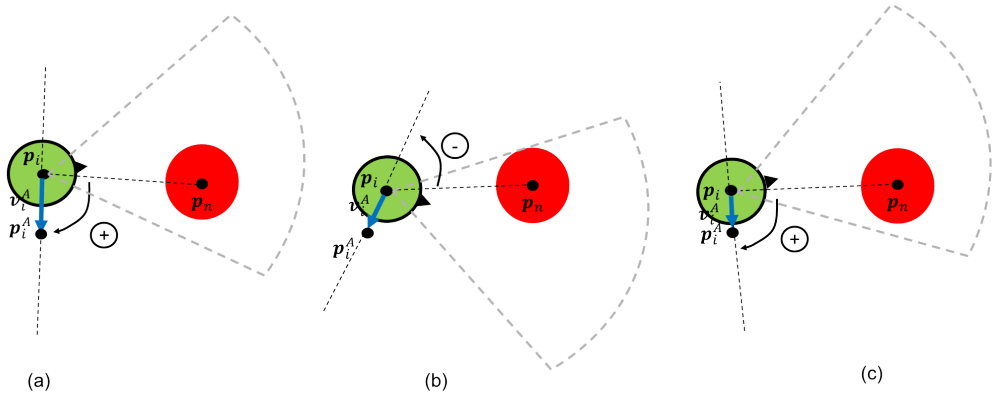


Figure 5.5: Illustration for agent's oscillating behaviours when the unit vector from agent c_i to its target o_n, $\mathbf{e}_{in}$, is almost orthogonal to the collision-free velocity $\mathbf{v}_i^A$.

The algorithm first compares $\left|\Delta\delta_{in}^{+}\right|$ and $\left|\Delta\delta_{in}^{-}\right|$ and selects a candidate movement direction that leads to a smaller deviation angle difference in ΔT. The algorithm then checks if the candidate movement direction results in a deviation angle, $|\delta_{in}(\Delta T)|$, that is larger than $\frac{\pi}{2}$, i.e. the agent heading opposite to its target. If yes, the movement direction is selected as the one resulting in a smaller $|\delta_{in}(\Delta T)|$. Otherwise, the candidate movement direction becomes the final movement direction. The robotic control vector is computed accordingly based on the movement direction. Algorithm 2 shows the details of the proposed heading-aware robotic control mapping algorithm.

5.3 Validation

This section validates the proposed method in terms of its view maintenance improvements via simulations. The proposed adaptive pair-wise responsibility sharing and heading-ware robotic control mapping algorithms (Proposed) is compared with the method that applies ORCA to differential-drive agents (ORCA-DD) [128], ORCA-DD with the proposed adaptive responsibility (ORCA-DD-AR) and ORCA-DD with the proposed heading-aware control mapping (ORCA-DD-HC). All methods are developed on top of the publicly available C++ library RVO2[2] (the implementation of [12]).

5.3.1 Experiment setup

Three scenarios without static obstacles are tested; Scenario I is a 30m $\times$ 30m square area with real people trajectories extracted from the PETS2009 dataset[3]. The trajectories of 10 people

[2]http://gamma.cs.unc.edu/RVO2. Last accessed: 30/08/2017
[3]http://www.cvg.reading.ac.uk/PETS2009. Last accessed: 30/08/2017

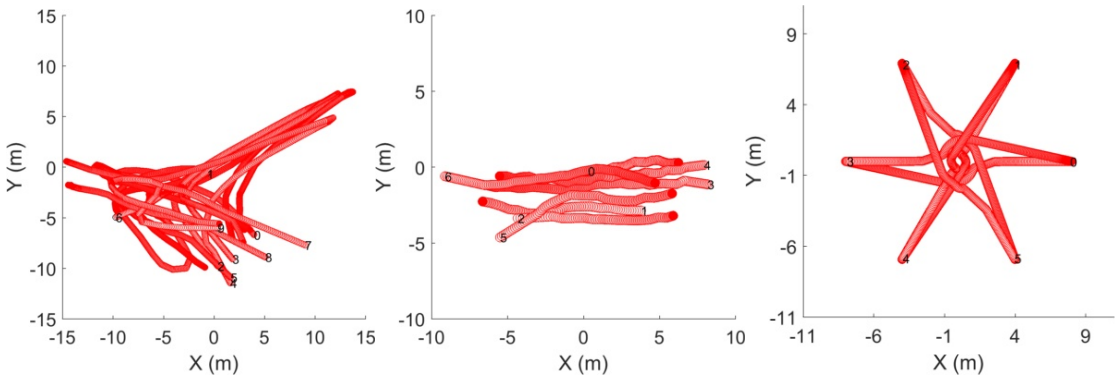


Figure 5.6: Trajectories of targets in three tested scenarios. Each target index (number in black) indicates the starting position of each target. All targets are indicated as a filled red circle with a radius of 0.3 m. The intensity of the red channel of each agent increases over time to distinguish the target positions at different time steps. (a) Scenario I with 10 trajectories of 60 s duration extracted from the PETS2009 S2L1 sequence. (b) Scenario II with 7 trajectories of 16 s duration extracted from the ETH2009 Walking Pedestrian Hotel sequence. (c) One setting of Scenarios III with six trajectories simulated using ORCA at a preferred speed of 1 m/s.

of 60 s duration is extracted from the S2L1 sequence. This sequence is chosen as it contains trajectories of people walking in a campus with various patterns, such as meeting from different positions, walking in pairs and walking back and forth (Fig. 5.6 (a)).

Scenario II is a 20m $\times$ 20m square area with real people trajectories extracted from the ETH Walking Pedestrian dataset[4]. We extract the trajectories of seven people of 16 s duration from the Hotel sequence. This sequence contains the trajectories of two groups of pedestrians intersecting with each other while moving at the opposite direction (Fig. 5.6 (b)).

The trajectories of the two above mentioned scenarios are extracted from the video sequence captured by a monocular camera. Some part of scene is not covered by the FoV of the camera. The positions of targets that do not enter the FoV in the beginning or exit the FoV earlier are interpolated with the last known position and velocity.

Scenario III is a 22m $\times$ 22m square area with simulated trajectories that are generated using ORCA [12]. Each simulated target is initialised on a circle and navigates towards its goal position which is set at the target's facing direction with a distance of 16m. The preferred speed of each target is set to 1 m/s. The number of targets in Scenario III varies from two to ten. An example scenario with six targets is shown in Fig. 5.6 (c).

Targets and agents are all modelled a circle of radius 0.3 m. The radius is enlarged to 0.6 m

[4]http://www.vision.ee.ethz.ch/en/datasets. Last accessed: 30/08/2017

when deriving the velocity constraints with ORCA, in order to compensate the trajectory tracking error [128]. In all scenarios, agents follow a differential-drive kinematic model. The state of an agent c_i at t is updated as:

$$\begin{bmatrix} x_i(t+1) \\ y_i(t+1) \\ \theta_i(t+1) \\ \dot{x_i}(t+1) \\ \dot{y_i}(t+1) \\ \dot{\theta_i}(t+1) \end{bmatrix} = \begin{bmatrix} x_i(t) + v_i(t)\Delta T \cos(\theta_i(t)) \\ y_i(t) + v_i(t)\Delta T \sin(\theta_i(t)) \\ \theta_i(t) + \omega_i(t)\Delta T \\ v_i(t)\cos(\theta_i(t)) \\ v_i(t)\sin(\theta_i(t)) \\ \omega_i(t), \end{bmatrix} \tag{5.14}$$

where the camera heading direction θ_i at any t is within $(-\pi, \pi]$. Each agent has a maximum speed $v_{\max} = 2$m/s, which is set to be larger than the speed of a target in order to guarantee the capture of the target within the camera's FoV. All targets are initialised at the agent's heading direction at the desired agent-target distance that is set to 2 m.

We use the ground-truth states of targets and agents without considering inaccuracy in the state estimation in the following experiments. More specifically, each agent knows the states of targets and agents within its avoidance range and knows the state of its own target at all time.

Setting the avoidance range too large makes each agent to account more agents and targets for collision avoidance, which leads to a higher chance of empty-set cases. It is also undesirable to set the avoidance range too small because agents may not have enough space for collision-free manoeuvre due to late aversion. We set the agent avoidance range to $2v_{\max}$ as it is the worst case for a collision between a pair of agents in one second.

Similar trade-off exists when setting the time horizon τ [11]. A larger τ allows for earlier aversion but sets more strict velocity constraints which can lead to empty-set cases. A smaller τ allows agents to have more time moving at their preferred velocity, which can be problematic when their target intersects with each other and leaves little room for collision-free manoeuvre. We set $\tau = 3$ for its moderate collision avoidance performance in all settings.

The on-board camera is assumed with a viewing angle $\phi = 90^\circ$. The view maintenance performance is decomposed into two aspects: the maintenance of deviation angle from the agent heading to its target and the maintenance of agent-target distance at the desired distance.

The deviation angle maintenance is measured by the ratio of time when the deviation angle

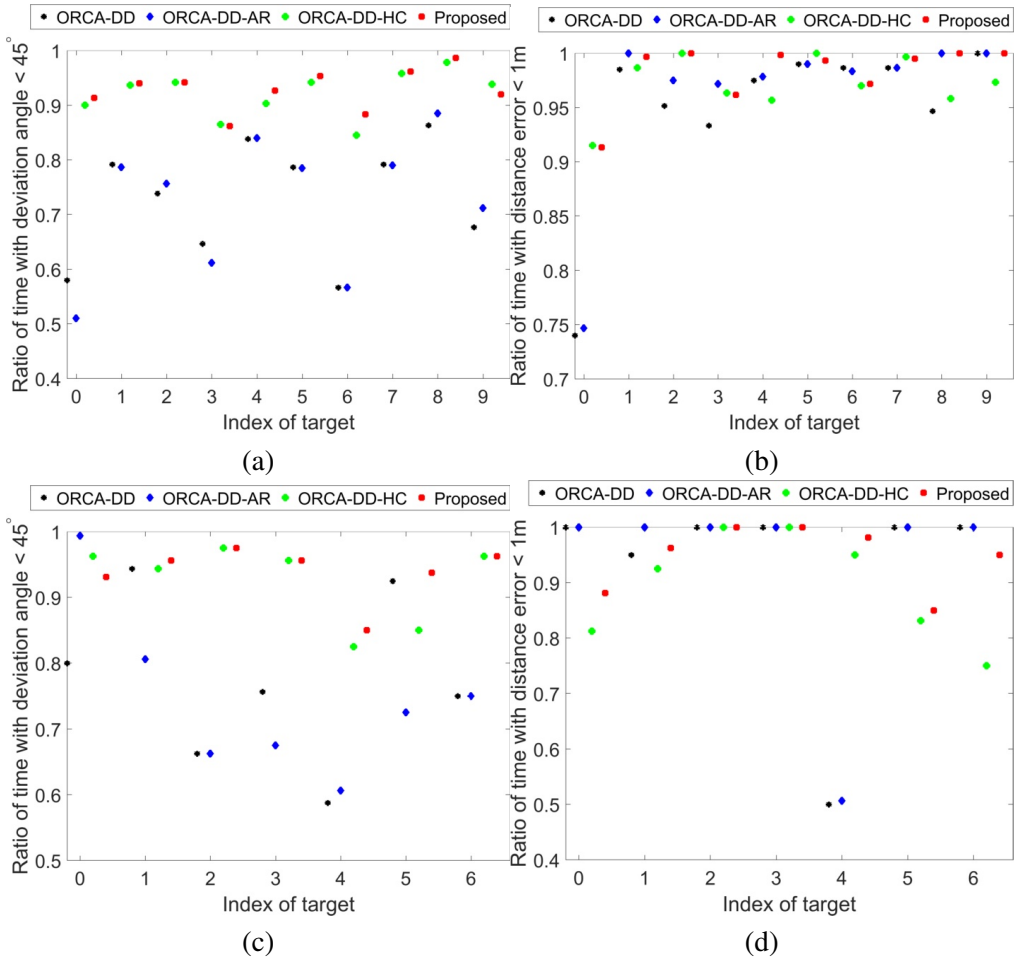


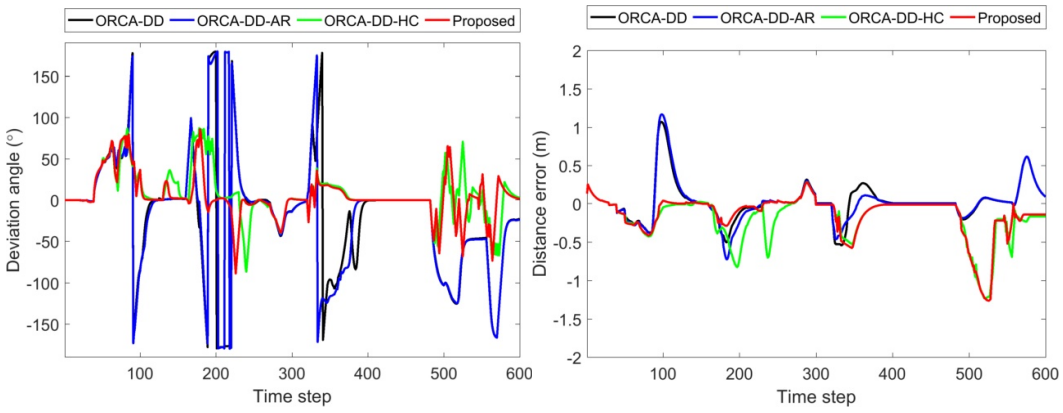
Figure 5.7: View maintenance performance of each agent. (a) The deviation angle maintenance ratio in Scenario I. (b) The distance maintenance ratio in Scenario I. (c) The deviation angle maintenance ratio in Scenario II. (d) The distance maintenance ratio in Scenario II.

from the agent heading towards its target is smaller than $\frac{\phi}{2} = 45°$:

$$\frac{1}{T}\sum_{t=1}^{T}|\delta_{in}(t)| < 45°, \tag{5.15}$$

where T is the experimental time steps. The distance maintenance measure is the ratio of time when the distance error between the agent-target distance and the desired distance is smaller than half of the desired distance, i.e. 1 m:

$$\frac{1}{T}\sum_{t=1}^{T}|d_{in}(t) - d_{in}^{*}| < 1. \tag{5.16}$$



(a) (b)

Figure 5.8: Results of the view maintenance performance over time of agent 6 in Scenario I. (a) Results of the deviation angle. (b) Results of the distance error.

5.3.2 Results discussion

Fig. 5.7 shows the view maintenance performance on each target in Scenario I and Scenario II. In general, the proposed adaptive responsibility sharing (ORCA-DD-AR) has the advantage over ORCA-DD in either maintaining deviation angle or agent-target distance. For example, in the case of target 3 in Scenario I, ORCA-DD-AR achieves a higher deviation angle maintenance ratio but a lower distance maintenance ratio compared to that achieved by ORCA-DD. Same behaviour is observed in Scenario II as well, e.g. target 1. The proposed heading-aware robotic control mapping algorithm shows the advantage in maintaining the agent heading towards its target for all targets in Scenario I and most targets in Scenario II. This is because the control mapping algorithm in ORCA-DD and ORCA-DD-AR makes agents only perform forward motion. This can easily cause the agent to head opposite to its target, i.e. the absolute value of the deviation angle is larger than 90°, when the collision-free velocity is backwards or the target moves back and forth. However ORCA-DD-HC cannot guarantee the advantages for all targets due to the highly dynamic scene created by multiple agents and targets, e.g. the agent that is actively tracking target 5 does not achieve a better deviation angle maintenance. When the heading-aware robotic control mapping algorithm is combined together with adaptive responsibility sharing (Proposed), the deviation angle maintenance performance can outperform other three methods for all targets in both Scenario I and Scenario II.

With ORCA-DD, ORCA-DD-AR, ORCA-DD-HC and the proposed method, the deviation angle maintenance ratio averaged over the number of targets is 0.77, 0.74, 0.92 and 0.94, respectively. The proposed method improves the deviation angle maintenance ratio by 21% compared

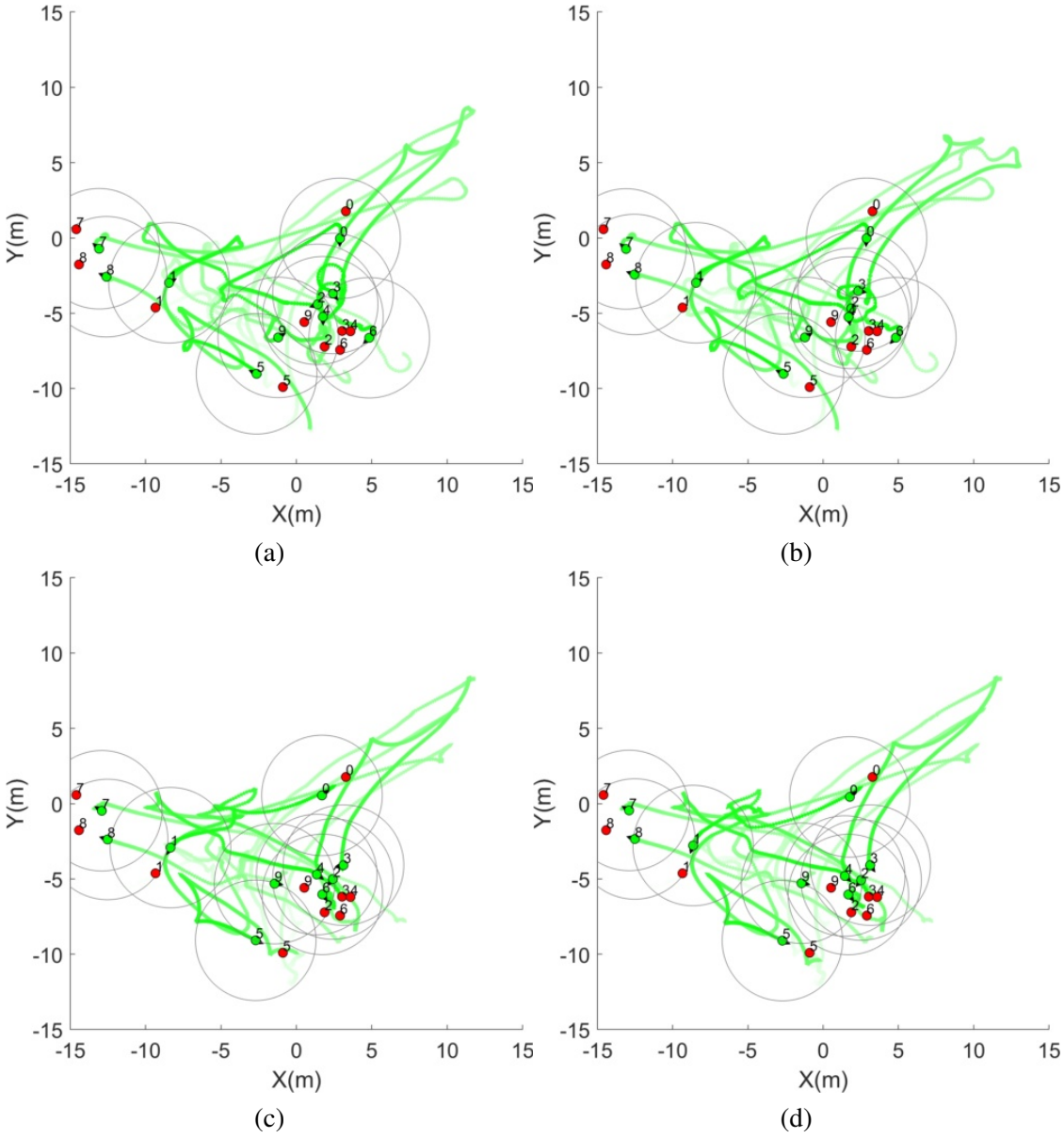


Figure 5.9: Resulted trajectories of 10 agents with each of them actively tracking a target (indicated as a red filled circle) in Scenario I. Each agent is represented as a green circular shape with a filled triangular to indicate the agent heading direction. Each agent is following its target with the same index. The trajectory of an agent is indicated by the green filled triangle (indicating the agent heading) with an increasing intensity of the green channel over time. The grey circle defines the area within the collision avoidance range of an agent. (a) Resulted trajectories with ORCA-DD. (b) Resulted trajectories with ORCA-DD-AR. (c) Resulted trajectories with ORCA-DD-HC. (d) Resulted trajectories with the proposed method.

to that of ORCA-DD in Scenario II. With ORCA-DD, ORCA-DD-AR, ORCA-DD-HC and the proposed method, the deviation angle maintenance ratio averaged over the number of targets is 0.73, 0.72, 0.92 and 0.93, respectively. The proposed method improves the deviation angle maintenance ratio by 27% compared to that of ORCA-DD in Scenario I. The adaptive responsibility

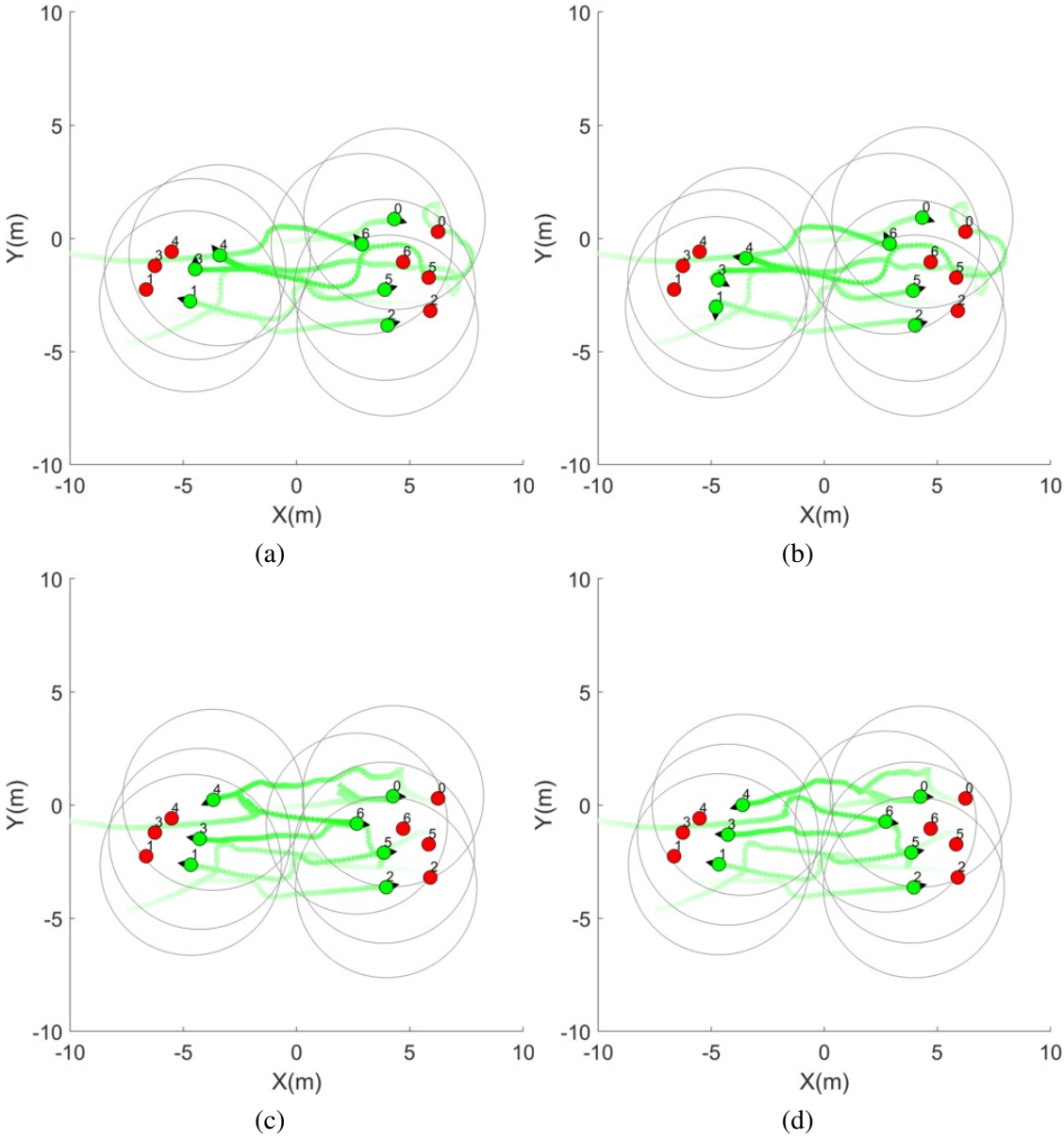
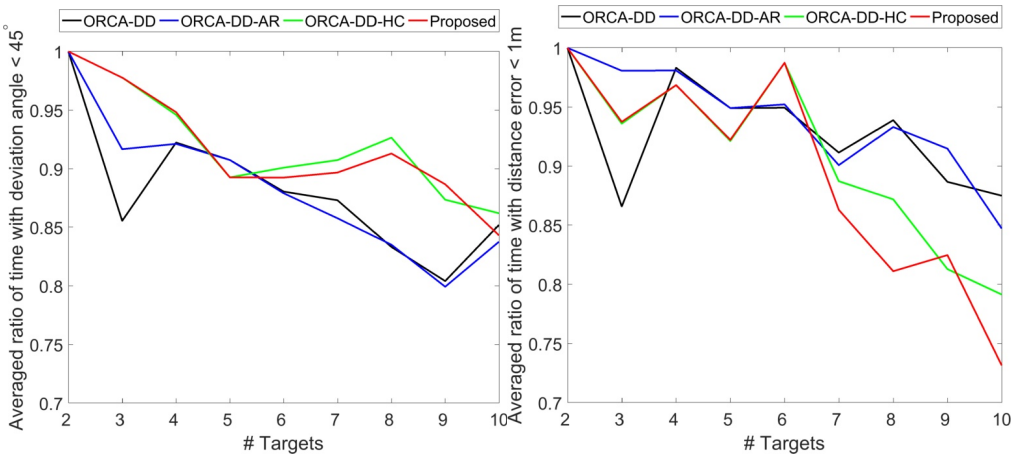


Figure 5.10: Resulted trajectories of seven agents with each of them actively tracking a target (indicated as a red filled circle) in Scenario II. Each agent is represented as a green circle with a filled triangle to indicate the agent heading direction. Each agent is following its target with the same index. The trajectory of an agent is indicated by the green filled triangle (indicating the agent heading) with increasing intensities of the green channel over time. The grey circle defines the area within the collision avoidance range of an agent. (a) Resulted trajectories with ORCA-DD. (b) Resulted trajectories with ORCA-DD-AR. (c) Resulted trajectories with ORCA-DD-HC. (d) Resulted trajectories with the proposed method.

algorithm is not advantageous in deviation angle maintenance while the heading-ware control mapping has a prominent advantage over ORCA-DD.

We also observe that the improvement on the deviation angle maintenance of Scenario I is higher than that of Scenario II. This is because trajectories in Scenario II (from ETH Walking



(a) (b)

Figure 5.11: Averaged view maintenance performance with increasing numbers of targets in Scenario III. (a) Angle maintenance ratio. (b) Distance maintenance ratio.

Pedestrian dataset) only have forward motion, i.e. people move at their facing direction without turning back, while trajectories in Scenario I contains also backward motion, i.e. people initially move at their facing direction and turn back afterwards. Such backward target motion can cause agents to move backward and potentially lose their target out of the FoV using ORCA-DD. The proposed heading-ware control mapping algorithm helps agents to move backward without heading opposite to their target.

Although the proposed method shows the advantage in maintaining deviation angle, it may come at the cost of a reduced distance maintenance performance, e.g. the case of agent 6 in Scenario I, as shown in Fig. 5.7(b). An agent with ORCA-DD can maintain the target at the desired distance but with the agent heading completely opposite to its target. Instead, the proposed heading-aware control algorithm forces the agent to head towards its target, which can cause an agent deviate from its desired agent-target distance due to the adjustments of agent heading. Fig. 5.8 shows the viewing maintenance performance of agent 6 in Scenario I over time. Agent 6 achieves a higher deviation angle maintenance ratio but a lower distance maintenance ratio using the proposed method, compared to that of ORCA-DD.

On average, the distance maintenance ratio achieved by ORCA-DD, ORCA-DD-AR, ORCA-DD-HC and the proposed method in Scenario I are 0.95, 0.96, 0.97 and 0.98, respectively. In Scenario II, the distance maintenance ratio achieved by ORCA-DD, ORCA-DD-AR, ORCA-DD-HC and the proposed method are 0.92, 0.93, 0.90 and 0.95, respectively. The adaptive responsibility algorithm achieves a slightly better distance maintenance performance in two sce-

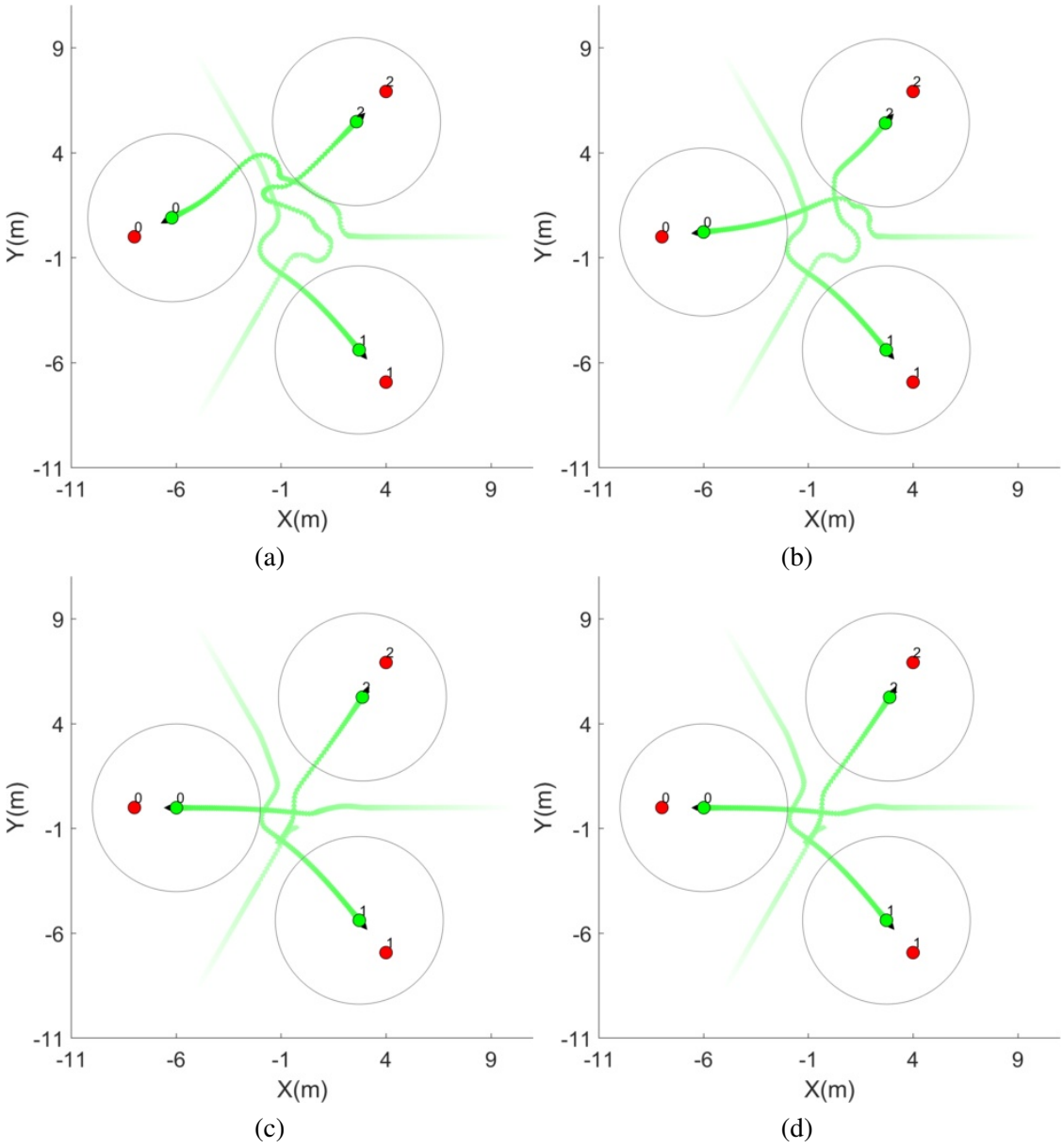


Figure 5.12: Resulted trajectories of three agents with each of them actively tracking a target (indicated as a red filled circle) in Scenario III. Each agent is represented as a green circular shape with a filled triangular to indicate the agent heading direction. The agent is following its target with the same index. The trajectory of an agent is indicated by the green filled triangle (indicating the agent heading) with an increasing intensity of the green channel over time. The grey circle defines the area within the collision avoidance range of an agent. (a) Resulted trajectories with ORCA-DD. (b) Resulted trajectories with ORCA-DD-AR. (c) Resulted trajectories with ORCA-DD-HC. (d) Resulted trajectories with the proposed method.

narios compare to that of ORCA-DD while the heading-aware control mapping algorithm does not show an obvious advantage over ORCA-DD. When both the adaptive responsibility algorithm and heading-aware control mapping algorithm are applied, a better distance maintenance performance can be achieved in both scenarios.

Fig. 5.9 shows the trajectories of all agents in Scenario I resulted from ORCA-DD (top left), ORCA-DD-AR (top right), ORCA-DD-HC (bottom left) and the proposed method (bottom right). At the end of the experiments, there are agents with ORCA-DD and ORCA-DD-AR (Fig. 5.9(a, b)) heading opposite to their target while the proposed heading-aware control mapping algorithm method (Fig. 5.9(c, d)) successfully maintains agents heading towards their target. Similar behaviours in terms of agent heading exist in Scenario II as well as shown in Fig. 5.10.

Fig. 5.11 shows the results of view maintenance performance with increasing numbers of targets in Scenario III. ORCA-DD-AR can achieve a better or equivalent view maintenance performance compared to that of ORCA-DD when there are less than seven targets. The three-agent case shows the advantage of the proposed method in maintaining both the deviation angle and agent-target distance. The resulted trajectories of three agents in Scenario III with different methods are shown in Fig. 5.12. The resulted trajectories with ORCA-DD-HC (Fig. 5.12(c)) are almost identical to the ones with the proposed method (Fig. 5.12(d)). This is because with the heading-aware control mapping, agents in this specific scenario do not require to share variant pair-wise responsibility as agents maintain the view on their target equally well.

When there are more targets intersecting simultaneously, agents are closely around each other which leaves a little space for agents to perform collision-free manoeuvre. All agents may encounter a high risk of losing their target. In such cases, the adaptive responsibility becomes less effective and can even worsen the view maintenance, for example, in the case of seven targets. In fact, when there is not enough free space for agents to move, ORCA cannot guarantee collision-free manoeuvres for each agent due to the strategy of handling empty-set cases: agents without any accessible collision-free control are allowed to intrude their velocity constraints to find at least one accessible velocity. The chances of collision are higher when agents has kinematic constraints and when agents are close to each other. In Scenario III, when there are eight targets intersecting simultaneously, agents with all methods may experience some collisions with other agents or targets. Therefore, the results for more than seven targets are not discussed.

In general, the proposed heading-aware control mapping algorithm can achieve a better deviation angle maintenance ratio in Scenario III. However, we notice that there are cases, e.g. with five targets, the proposed ORCA-DD-HC achieves a slightly smaller deviation angle maintenance ratio than that of ORCA-DD and ORCA-DD-AR. This is because in Scenario III, targets move forward to their goal position at all time and the collision-free velocity may only cause the agent

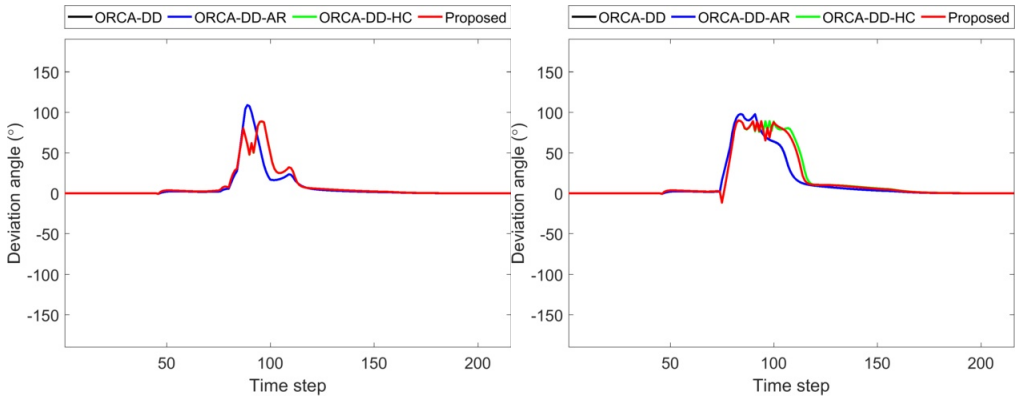


Figure 5.13: The deviation angle of two agents to their target over time in Scenario III with five targets. The proposed method adjusts the agent heading towards its target and causes a slightly longer time when the deviation angle is larger than half of the camera's view angle. (a) Results of agent 1. (b) Results of agent 2.

to head opposite to its target for a short time. In such cases, the proposed method attempts to adjust the agent heading towards its target which leads to a slightly longer time when the deviation angle is larger than half of the FoV view angle (see Fig. 5.13).

5.4 Summary

This Chapter proposed an ORCA-based collision avoidance method for active visual tracking in multi-agent scenarios. To address view maintenance during collision avoidance manoeuvres, we proposed an adaptive pair-wise responsibility sharing algorithm based on the velocity difference between the preferred velocity and the current velocity of an agent. When mapping the collision-free velocity to a feasible robotic control, we further proposed the heading-aware control mapping algorithm that accounts for the agent heading deviation from its target in a smooth manner.

We evaluated the proposed two elements both independently and jointly using real people trajectories that are extracted from public datasets, and compared them with an ORCA-based method for differential-drive robotic platforms (ORCA-DD) [128]. The adaptive pair-wise responsibility algorithm demonstrated a slightly better distance maintenance performance, while the heading-aware control mapping algorithm showed a prominent advantage in the deviation angle maintenance over the compared method. When two elements are jointly applied, the proposed method achieved $\geq 20\%$ improvement in the deviation angle maintenance and 3% improvement in distance maintenance, compared to ORCA-DD with the tested trajectories. However the ad-

vantage in view maintenance of the proposed method can not be guaranteed for all targets at all time due to the high dynamics in multi-agent scenes. The proposed adaptive responsibility sharing algorithm may lose its advantage in view maintenance when many targets, i.e. seven targets, are intersecting simultaneously. As future work, we plan to further investigate the real time and collision avoidance performance with multiple robotic platforms using Robotic Operating System.

Chapter 6

Robotic validation of multi-agent active visual following

6.1 Overview

In this Chapter, we propose and implement a scalable and fully distributed multi-agent system for concurrent active visual following using edge processing. At each cycle, each agent estimates the states of all agents and targets via distributed tracking with static cameras, and derives its own collision-free robotic control on board using the method proposed in Chapter 5. We design and implement the system on Robotic Operating System (ROS) and address challenges such as wireless communication imperfections, including packet delay and loss, caused by interferences and concurrent transmission, and tracking failures caused by detection or transmission errors [C5]. With ceiling-mounted cameras and Kobuki robotic platforms, we validate the proposed system in terms of view maintenance, collision avoidance and system latency.

This Chapter is organised as follows. Section 6.2 introduces the proposed infrastructure on ROS, where we detail the implementation of each functionality in subsections. Section 6.3 validates the implemented system with four robotic platforms and three web cameras. Subsection 6.3.1 describes the experimental setup and Subsection 6.3.2 introduces the camera calibration that is an essential preparation work prior to multi-camera experiments. Subsection 6.3.4 presents the evaluation of real-time performance of the distributed tracking system. Subsection 6.3.3 presents the motion control performance with the method in Chapter 5 under two scenarios. Finally, conclusion is drawn in Section 6.4.

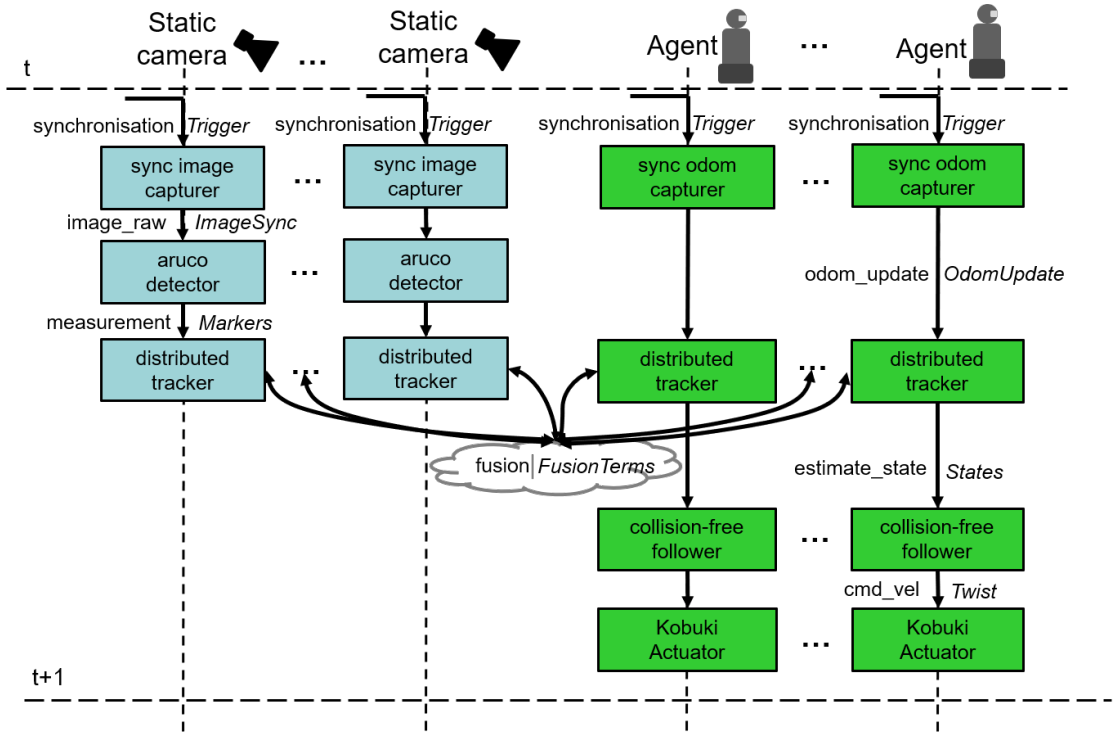


Figure 6.1: ROS nodes and messages for static cameras (blue) and agents (green). The left-hand side of an arrow indicates the name of a ROS topic. The right-hand side is for the type of a ROS message communicated on the ROS topic.

6.2 ROS implementation

We assume that all static cameras know the targets and agents to track, and agents know their respective target to follow.

At each cycle, static cameras detect targets and agents and estimate their pose from visual detections, while agents estimate their pose using on-board odometry sensors (e.g. wheel encoders and imu sensors). Agents perform local filtering using the Extended Information Filter (EIF) and distributed fusion with neighbouring static cameras by exchanging local state estimates. Finally, each agent accounts for view maintenance as an additional constraint to compute its collision-free motion control with ORCA.

Each static camera and agent is implemented as a composition of ROS nodes and ROS message flows in Fig. 6.1. Each ROS node is essentially an executable file that supports event-driven programming and each ROS message is a data structure that comprises typed data fields. ROS nodes interact with each other by subscribing from or publishing to ROS topics. In our implementation, the *synchroniser* node provides a common time reference for synchronised sensor capturing by broadcasting Trigger messages at a fixed frequency.

For each static camera, the *sync image capturer* node is used for synchronised image capture on receiving the *Trigger* message. On receiving an *ImageSync* message, the static camera detects targets and agents on the image plane. To simplify the detection procedure, the *aruco detector* node detects the fiducial markers attached on top of the robotic platforms and estimates the poses of detected markers. The *aruco detector* node finally encapsulates the pose measurements in the *Markers* message and publish it to the *measurement* topic. Finally, on receiving the *Markers* message, the *distributed tracker* node performs local updates and publishes the *FusionTerms* messages to the shared *fusion* topic. On hearing the *FusionTerms* messages from neighbours, the node performs distributed fusion.

For each agent, the *sync odom capturer* node captures synchronised odom measurement and updates the local state on receiving the *Trigger* message. On receiving the *OdomUpdate* message that carries the updated pose of the agent itself, the *distributed tracker* node performs local filtering and publishes the *FusionTerms* messages to the shared *fusion* topic, which is then subscribed by the node itself to perform distributed fusion with neighbours. The *distributed tracker* finally publishes the *States* message that encapsulates the estimated poses of all agents and targets to the *estimate_state* topic. On receiving the *States* message, the *collision-free follower* node finally computes the collision-free robotic control, which is then sent as the *Twist* message to the *Kobuki actuator* node for control execution.

In the following sections, we will present in details the algorithm implementation at each node and the data structure of each message.

6.2.1 Synchronised sensor capturing

Time-stamping sensed data simultaneously is essential for distributed tracking. We synchronise the clocks across multiple machines in the local network using the Network Time Protocol with a machine coordinating this process (detailed procedure can be found here [1]). Although the time discrepancy among machines can be kept within 10^{-3} seconds after a few minutes, there is a noticeable delay caused by the initialisation of the nodes in each machine.

We address this delay in initialisation by triggering the sensor-data capturing at each machine by listening to the synchronisation machine that runs the *synchroniser* node. The *synchroniser* node provides a common time reference by broadcasting *Trigger* messages at a fixed frequency. The *synchroniser* node first delays its processing to wait for all nodes to be initialised at each

[1] http://wiki.ros.org/ROS/NetworkSetup Last accessed: 30/08/2017

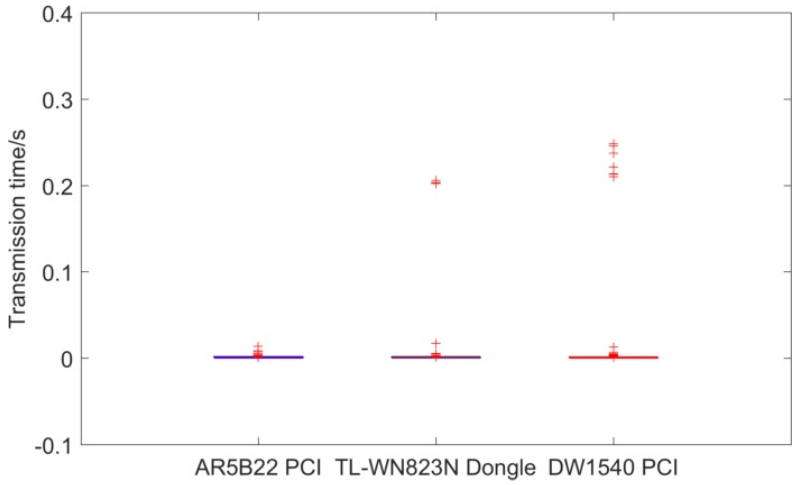


Figure 6.2: Transmission times for trigger messages with different WiFi devices.

machine and then sends *Trigger* messages through the *synchronisation* topic. This delay should increase with the number of machines and the number of nodes involved. The *Trigger* message encapsulates a ROS *Header* message and the frequency of transmission. The ROS *Header* message [2] contains typed data fields for the sequence number and the time stamp whose values are set by ROS middleware automatically.

Both the *sync image capturer* node at static cameras and the *sync odom capturer* node at agents listen to the *synchronisation* topic, while capturing the sensory data at a higher frequency compared to the system frequency. This makes use of *cv_camera* for static cameras and *robot_pose_ekf* for agents. The main idea is that on receiving the *Trigger* message, the *sync image capturer* node and the *sync odom capturer* node encapsulate and publish the sensory data for which the time stamp is closest to the time stamp of the received *Trigger* message.

However, as transmission delays vary with different WiFi devices and conditions (e.g. environmental interferences), the machines cannot rely only on the receipt of *Trigger* messages. Fig. 6.2 shows as an example of the transmission time for the *Trigger* messages from one machine to another with different WiFi devices.

To address the variable transmission delay, we enable on each machine a local timer whose duration is between two consecutive *Trigger* messages. Each machine performs synchronised sensor capturing every time the timer is reset. The local timer is reset when the machine receives an on-time *Trigger* message, so that it can be maintained consistent with the synchronisation machine, or when the local timer expires if the *Trigger* messages are not received on time.

[2] http://wiki.ros.org/Messages

Once the sensory data is obtained, the *sync image capturer* node at static cameras publishes *ImageSync* messages to the *image_raw* topic. The *ImageSync* message encapsulates the image data as a ROS *Image* message and the sequence number of the capture needed to fuse information from multiple machines. Similarly, the *sync odom capturer* node at agents publishes *OdomUpdate* messages to the *odom_update* topic. The *OdomUpdate* message encapsulates the estimated agent pose in the form of ROS *PoseWithCovariance* messages and the sequence number of the capture.

6.2.2 Target and agent detection

To facilitate the detection process and to focus our analysis on the distributed tracking and collision avoidance processes, we use fiducial markers. Specifically, we use the ArUco marker and its detector [44] for its robustness to false positives and availability in existing libraries, e.g. OpenCV and ROS. The detector uses a marker dictionary and outputs the corners and the IDs of the markers. While the ArUco marker detector is robust to false positives via shape filtering and code verification [44], it suffers from false negatives under illumination changes or sudden movements.

To address this problem we use a KLT keypoint tracker on the corners of the marker. In the case of successful marker detection, the keypoints are updated by the corners of the detected marker, otherwise the marker detection will use the corners tracked by the KLT tracker.

Although the measurements obtained from the KLT tracker are given lower confidence than those from the marker detector, the use of the KLT tracker reduces the number of missing measurements and therefore helps the state estimation process of the distributed tracker.

Finally, we estimate the poses of detected markers in world coordinate using the camera parameters obtained from calibration (We detail how to obtain the camera parameters in Subsection 6.3.2).

The *aruco dectctor* node implements the above described procedure. On receiving an *ImageSync* message, the node performs detection for markers detection and pose estimation, and finally publishes a *Markers* message to the *measurement* topic. The *Markers* message consists of a ROS *Header* message, the sequence number, and an array of *Marker* messages. Each *Marker* message encapsulates the marker ID, a boolean to indicate whether the marker is detected and a *PoseWithCovariance* message for the estimated pose. For the markers which are not detected, we set the boolean to false and leave the pose as NULL.

6.2.3 Distributed tracking

Static cameras detect targets and agents, while agents contribute the measurements of their own pose obtained from on-board odometry.

Each c_i first updates its local state estimate using EIF [62]. Let $\mathbf{y}_i^w(t)$ be the information form of the local state estimate at c_i for target o_n (when $w = n$) or agent j (when $w = j$). Let $\mathbf{Y}_i^w(t)$ be the information matrix corresponding to $\mathbf{y}_i^w(t)$. $\mathbf{y}_i^w(t)$ and $\mathbf{Y}_i^w(t)$ are obtained using Eq. 2.8.

Each c_i then performs distributed fusion by iteratively exchanging local updates with neighbours. We employ the Iterative Covariance Intersection (ICI) for distributed fusion, as it does not require knowledge of network connectivity [53, 61], which is desired for dynamic networks with mobile agents. Let $\mathbf{x}_i^{w,k}(t)$ and $\mathbf{X}_i^{w,k}(t)$ denote the fusion terms that c_i exchanges with neighbours and k be the index of fusion iteration. The initial terms are computed as in Eq. 2.13.

Each c_i exchanges with neighbours the terms of all targets and agents. At each iteration k, the terms are updated similarly as in Eq. 2.14:

$$\begin{aligned}\mathbf{x}_i^{w,k}(t) &= \sum_{c_j \in \mathbf{C}_i^R(t)} \mu_j^{w,k}(t)\mathbf{x}_j^{w,k-1}(t) \\ \mathbf{X}_i^{w,k}(t) &= \sum_{c_j \in \mathbf{C}_i^R(t)} \mu_j^{w,k}(t)\mathbf{X}_j^{w,k-1}(t),\end{aligned} \tag{6.1}$$

where $\mathbf{C}_i^R(t)$ is the set of neighbouring static cameras and agents whose terms are successfully received by c_i; and $\mu_j^{w,k}(t)$ is the weight assigned to the information from c_j at iteration k. We compute the weight similarly as in Eq. 2.15:

$$\mu_i^{w,k}(t) = \frac{tr(\mathbf{X}_i^{w,k-1}(t))}{tr(\mathbf{X}^{w,k-1}(t))}, \tag{6.2}$$

where $tr(\cdot)$ is the trace of a matrix and

$$\mathbf{X}^{w,k-1} = \sum_{c_j \in \mathbf{C}_i^R(t)} tr(\mathbf{X}_j^{w,k-1}(t)). \tag{6.3}$$

For fully connected networks, only one round of fusion suffices to converge, while for not fully connected networks, the number of rounds of fusion depends on the level of network connectivity.

The *distributed tracker* node implements the distributed tracking fusion. On receiving the

Markers message, the *distributed tracker* node at static cameras performs local updates and publishes the *FusionTerms* messages to the *fusion* topic. Similarly for agents, on receiving the *OdomUpdate* message, the *distributed tracker* node performs local filtering and publishes the *FusionTerms* messages to the *fusion* topic. The *FusionTerms* message consists of the *Header* message, the source ID to indicate the machine that sends the message, the iteration number, the sequence number and an array of the *FusionTerm* messages. The *FusionTerm* message is composed of the marker ID, the boolean to indicates if the marker is detected, an array for information vector and an array for information matrix.

The nodes at all static cameras and agents subscribe to the same *fusion* topic in order to perform distributed fusion with neighbours. The fusion process of *distributed tracker* node at an agent is the same as that at a static camera. On receiving a *FusionTerms* message, the node temporarily stores the terms in a local cache for each neighbour. The node performs one fusion iteration when the *FusionTerms* messages from all neighbours are received or a pre-set timer expires. The timer is necessary due to the presence of packer loss and delay, which can stop the fusion when the node cannot receive all *FusionTerms* messages. We set the timer to 0.15 s experimentally. The node currently performs one round of fusion due to the full connectivity.

At the end of fusion, the node at agents prepares the fused posterior state estimates into the *States* message and publishes it to the *estimate_state* topic. The *States* message encapsulates the *Header* message, the sequence number, the boolean that indicates if the agent has the estimated states of all agents and targets and an array of *State* messages which carries the estimated state of each agent and target. The *State* message has the ID of agent or target, the boolean that indicates if the estimated state is available and the *Pose2D* message for the estimated state in 2D.

As the communication is lossy and static cameras may fail to provide accurate measurements, if a node has no measurement of an agent or a target provided by any static cameras for 10 consecutive time steps, the node will reinitialise the local filter and will not provide the states in the *States* message by setting the boolean to false. Note that the measurements from odom sensors are with the initial coordinate of the agent instead of the common world coordinate. The *distributed tracker* node at agents will not join fusion with static cameras until its local filer is initialised by the information from static cameras.

6.2.4 Collision-free motion control

Each agent computes its preferred velocity and that of all neighbouring agents based on the estimated states of all targets and agents, and then derives the pair-wise velocity constraints induced by each of its neighbouring agents and targets. Agents that aim to maintain a target in their FoV have to account also the view maintenance constraint during collision avoidance manoeuvres. We achieve view maintenance by adapting the pair-wise responsibility when deriving the velocity constraints and by setting the feedback errors when computing the robotic control from the collision-free velocity with the objective of minimising the deviation angle of the agent's heading direction from its target. The details of the proposed algorithm are referred to Section 5.2 in Chapter 5.

The *collision-free follower* node implements the computation of the collision-free robotic control. On receiving the *States* message, the node first checks if the states of all agents and targets are available in the received message to proceed, otherwise the node stops the robotic platform. When it proceeds, the node computes and encapsulates the robotic control, i.e. speed and steering angle, in the *Twist* message that is published to the *cmd_vel* topic. On receiving the *Twist* message, the *Kobuki actuator* node derives the speeds for the two wheels, and actuates the robotic platform.

6.3 Experiment

In this section, we validate the proposed multi-agent system with four Kobuki Turtlebot platforms[3]: two platforms play the role of agents that actively track and follow their respective targets, and two platforms play the role of targets. We first assess the view maintenance and collision avoidance capability and then investigate latencies and real-time performance.

6.3.1 Experimental setup

Three ceiling-mounted static cameras are set to cover a 3 m $\times$ 3 m square indoor area where in our case is a teaching room (see Fig. 6.3(a)). The camera poses are not strictly constrained as long as the cameras share a partial field of view and the three cameras cover the whole experimental area. Each *agent Kobuki* moves in the area with the objective of tracking its own target while avoiding other agents and targets. Each *target Kobuki* moves along a predefined path at 0.1 m/s.

[3]http://kobuki.yujinrobot.com/. Last accessed: 28/02/2018

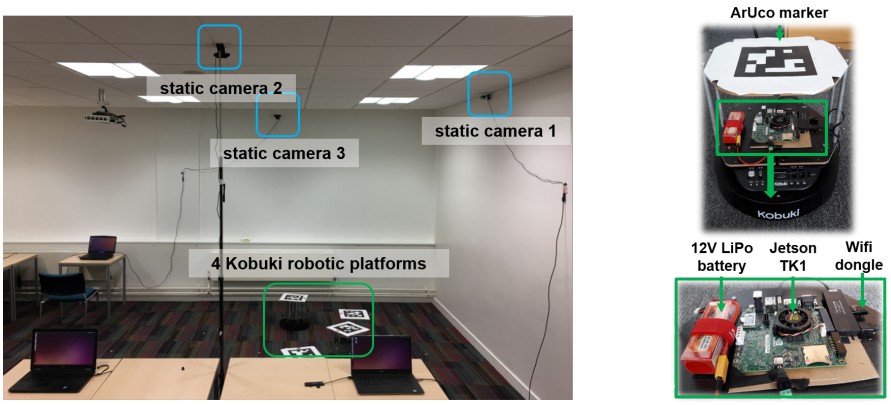


Figure 6.3: Experimental setup. (a) Blue circles indicate the positions of the static cameras. (b) A Kobuki with an additional computing unit (Jetson TK1 board supported by a 12V battery), a WiFi dongle and an ArUco marker. The agents communicate with neighbouring cameras and agents in an adhoc network.

This speed is slower than typical walking speed due to the constrained experimental area. The agent-target assignment is known as a priori and the desired agent-target distance is set to 0.8 m.

Each static camera is composed of a Logitech C920 HD Pro with a view angle 78° and image size of 1280×720, a processing unit (provided by a laptop) and a wireless module that supports the IEEE 802.11 protocol in adhoc mode. The three laptops are a Dell XPS laptop with core i7 (camera 1), a Dell Latitude E5550 with core i5 (camera 2) and a Dell Alienware laptop with core i5 (camera 3). The wireless device is either a PCI card or a USB dongle.

We added to each Kobuki a wireless module and an Nvidia Jetson TK1 with ARM quad-core Cortex-A15 as processing unit (see Fig. 6.3(b)). Each Kobuki follows differential-drive kinematics with two powered wheels, which can be controlled independently, and a passive caster wheel to maintain balance. The Kobuki has access to the odom measurements from both wheels and the IMU sensors. On top of each Kobuki, we place an ArUco marker with an ID to ease detection and identification for three *ambient cameras* (see Fig. 6.3(a)).

6.3.2 Camera calibration

Prior to the experiment discussion, we explain the procedure for camera calibration that is essential for achieving pose estimation with respect to the common world coordinate. Camera calibration estimates the camera parameters, i.e. intrinsics, distortion coefficients, and extrinsics based on a pinhole camera model. Calibration is essential to achieve the mapping between

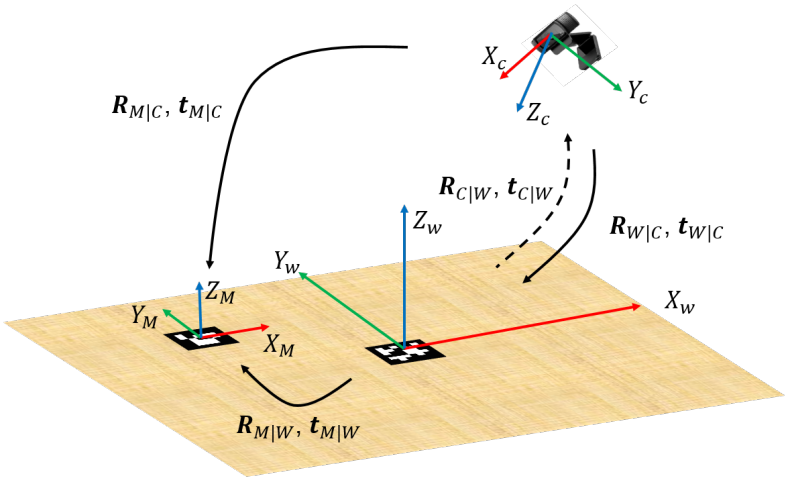


Figure 6.4: Illustration of how to transform the marker pose in the camera coordinate to a common reference coordinate.

image-plane detections to world coordinate. The intrinsics relate to the camera focal length, scaling factors for row pixels and column pixels and skew factor. The distortion coefficients relate to the lens imperfections, such as the pin-cushion effect. We use the ROS camera calibration tool to estimate the intrinsics and distortion coefficients, as it provides online frame selection to ensure sufficient variety of the checker board positions.

The extrinsics are the rotation and translation of a camera in world coordinate, and varies when the camera pose changes. The available tools for pose estimation provide the marker pose in its own camera coordinate rather than in world coordinate, we therefore need the knowledge of the camera pose in world coordinate as a prior, i.e. the extrinsics.

We obtain the camera extrinsics using a reference marker, whose coordinate is considered as world coordinate. Fig. 6.4 shows the calibration process for extrinsics. All rotation matrices, $\mathbf{R}$, are of size 3×3 and translation matrices, $\mathbf{t}$ are of size 3×1. W, C and M in the subscript of each rotation and translation matrix is short for World, Camera and Marker, respectively. The transformation is from the coordinate behind the vertical slash to the coordinate before the vertical slash. For example, $\mathbf{R}_{W|C}$ is the rotation matrix from camera coordinate to world coordinate. In order to obtain the marker rotation and translation from world coordinate to marker coordinate ($\mathbf{R}_{M|W}$ and $\mathbf{t}_{M|W}$), one can first get the transformation from world coordinate to camera coordinate ($\mathbf{R}_{C|W}$ and $\mathbf{t}_{C|W}$), and then from camera coordinate to marker coordinate ($\mathbf{R}_{M|C}$ and $\mathbf{t}_{M|C}$).

The transformation from world coordinate to camera coordinate ($\mathbf{R}_{C|W}$ and $\mathbf{t}_{C|W}$) is the reverse of transformation from camera coordinate to world coordinate ($\mathbf{R}_{C|W}$ and $\mathbf{t}_{C|W}$), computed

(a) (b)

Figure 6.5: Sample initial positions of targets and agents. Agents can be initialised at any positions and heading directions within the experimental area. (a) Scenario I. (b) Scenario II. The paths of target 1 and target 2 are shown as dashed lines in blue and in yellow, respectively.

as:

$$\begin{aligned} \mathbf{R}_{C|W} &= \mathbf{R}_{W|C}^{T} \\ \mathbf{t}_{C|W} &= -\mathbf{R}_{W|C}^{T}\mathbf{t}_{W|C}, \end{aligned} \tag{6.4}$$

Where $\mathbf{R}_{W|C}$ and $\mathbf{t}_{W|C}$ is the estimated rotation and translation of the reference marker. The extrinsics are the average of results estimated from successful marker detections, in order to be more robust to detection noises.

With the known camera pose in world coordinate ($\mathbf{R}_{C|W}$ and $\mathbf{t}_{C|W}$), we can transform the pose of a marker in camera coordinate to world coordinate:

$$\begin{aligned} \mathbf{R}_{M|W} &= \mathbf{R}_{M|C}\mathbf{R}_{C|W} \\ \mathbf{t}_{M|W} &= \mathbf{R}_{M|C}\mathbf{t}_{C|W} + \mathbf{t}_{M|C}, \end{aligned} \tag{6.5}$$

With the known camera pose in world coordinate ($\mathbf{R}_{C|W}$ and $\mathbf{t}_{C|W}$), we can transform the pose of a marker in camera coordinate to world coordinate:

$$\begin{aligned} \mathbf{R}_{M|W} &= \mathbf{R}_{M|C}\mathbf{R}_{C|W} \\ \mathbf{t}_{M|W} &= \mathbf{R}_{M|C}\mathbf{t}_{C|W} + \mathbf{t}_{M|C}, \end{aligned} \tag{6.6}$$

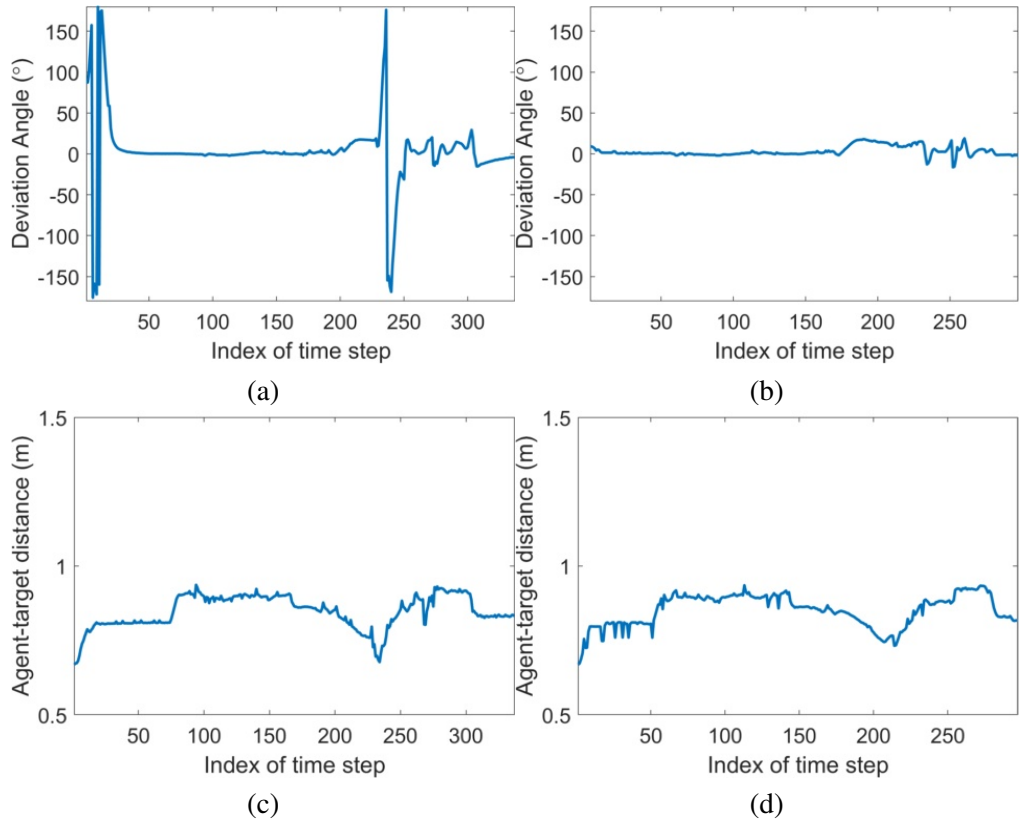


Figure 6.6: View maintenance performance of agent 2 with baseline and proposed method. Deviation angle of the agent from its heading to the target with (a) baseline method and (b) proposed method. Agent-target distance with (c) baseline method and (d) proposed method.

6.3.3 View maintenance and collision avoidance

In this section, we perform multi-robot experiment to demonstrate the advances of view maintenance using the proposed algorithm in Chapter 5 compared to the baseline method [128]. We quantify the view maintenance by the deviation angle of the agent heading to its target and the agent-target distance [128].

We consider two scenarios (Fig. 6.5): Scenario I is an intersection case and Scenario II is a meeting case. In both scenarios, agent 1 follows target 1 and agent 2 follows target 2. In Scenario I, the two targets are initialised with opposing directions along the diagonal of the experimental area. Targets move forward together and intersect with collision-free paths. Afterwards, the two targets turn backwards and intersect again by moving forward along straight lines. In Scenario II, the two targets are initialised with opposing directions along one side of the experimental area. Targets move towards each other and then turn 90° in order to move together (forth and back) facing the same direction. The positions and heading directions of agents can be initialised in a flexible manner within the experimental area.

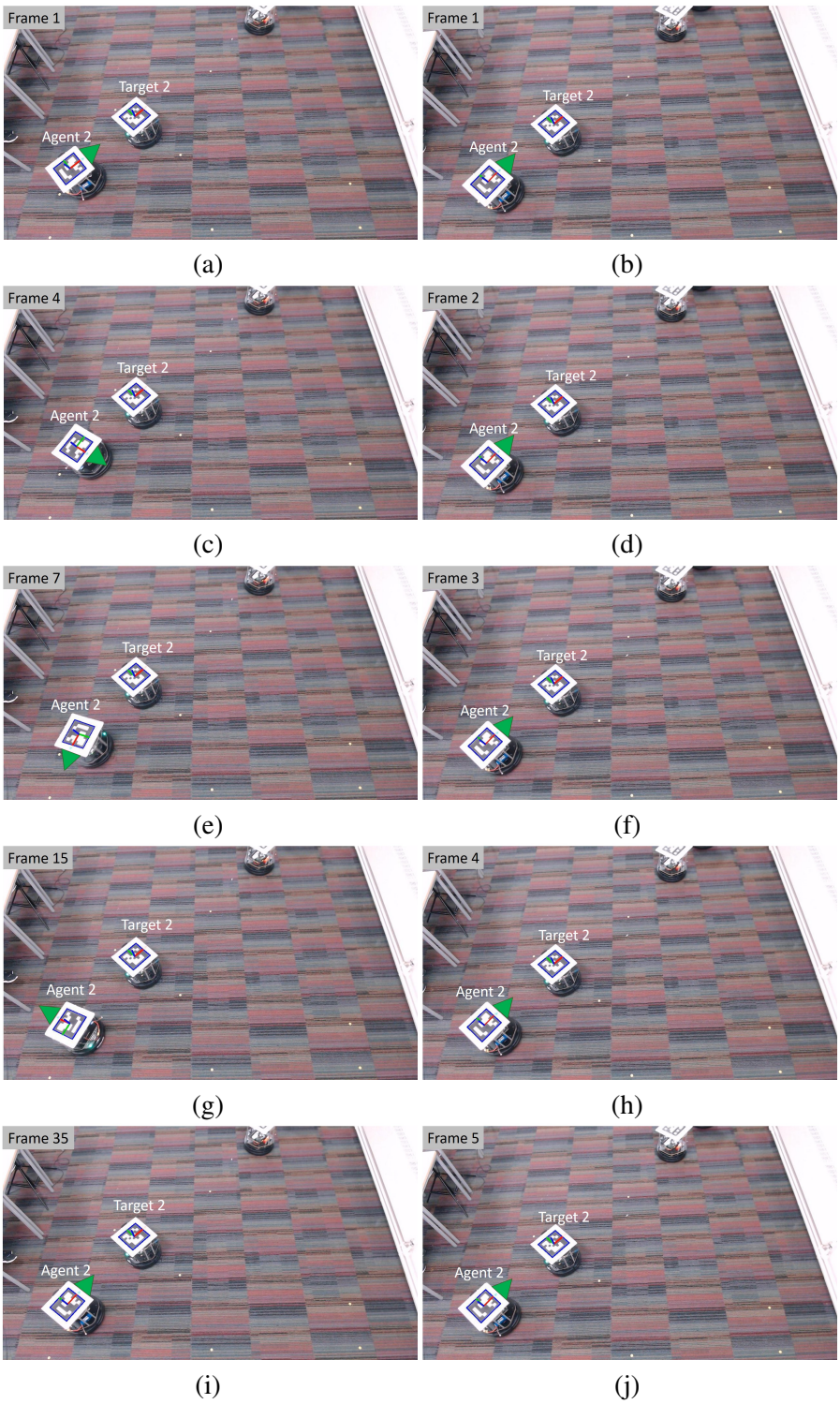


Figure 6.7: Sample frames from static camera 1 when agent 2 needs to move backwards with the baseline method (left column) and the proposed method (right column). The green triangle highlights the heading direction of each agent. Markers with blue bounding boxes are detected by the detector and markers with purple boxes are detected by the KLT corner tracker.

Fig. 6.6 compares the view maintenance of agent 2 in Scenario I with the baseline and the proposed method. Agent 2 is initialised at a distance that is smaller than the desired agent-target

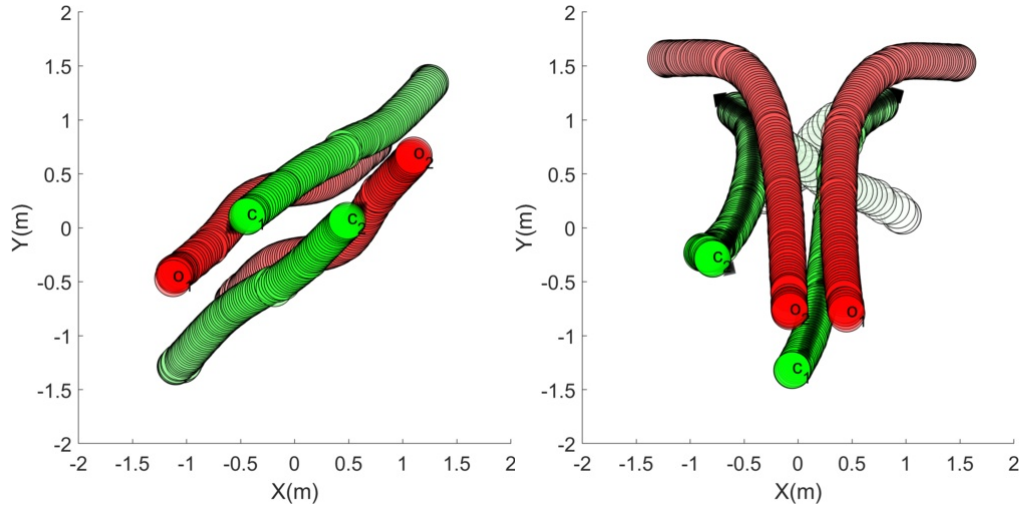


Figure 6.8: The paths of agents (green) and targets (red) in (a) Scenario I and (b) Scenario II. Agent c_1 (c_2) follows target o_1 (o_2).

distance, which leads to a backward velocity before target 2 starts to move. Agent 2 with the baseline method moves forward while turning to the back, which temporarily causes agent 2 to face the opposite direction to target 2, i.e. the absolute value of the deviation angle becomes 180°. With the proposed method, agent 2 moves backward without turning back, so that it can always face its target. As a result, the overall paths of agents with the proposed method is shorter than that with the baseline method. Fig. 6.7 shows some sample frames from static camera 1 to demonstrate the movements of agent 2 in the case of backward velocities with the baseline method (left column) and the proposed method (right column).

Fig. 6.8 shows the paths of agents and targets under both scenarios. There is no collision among agents and targets throughout the experiments. Fig. 6.9 shows sample frames from static camera 1 in Scenario II. Two agents need to move towards their target where potential collisions can occur if no avoidance manoeuvres are applied. The frames show that agents manage to avoid each other using the proposed ORCA-based method. The multi-camera tracking system extends the coverage and is robust to single-camera failures. Fig. 6.10 shows sample frames from static camera 1 and static camera 3, where we can observe that the system keeps tracking when one camera experiences missing or inaccurate detections.

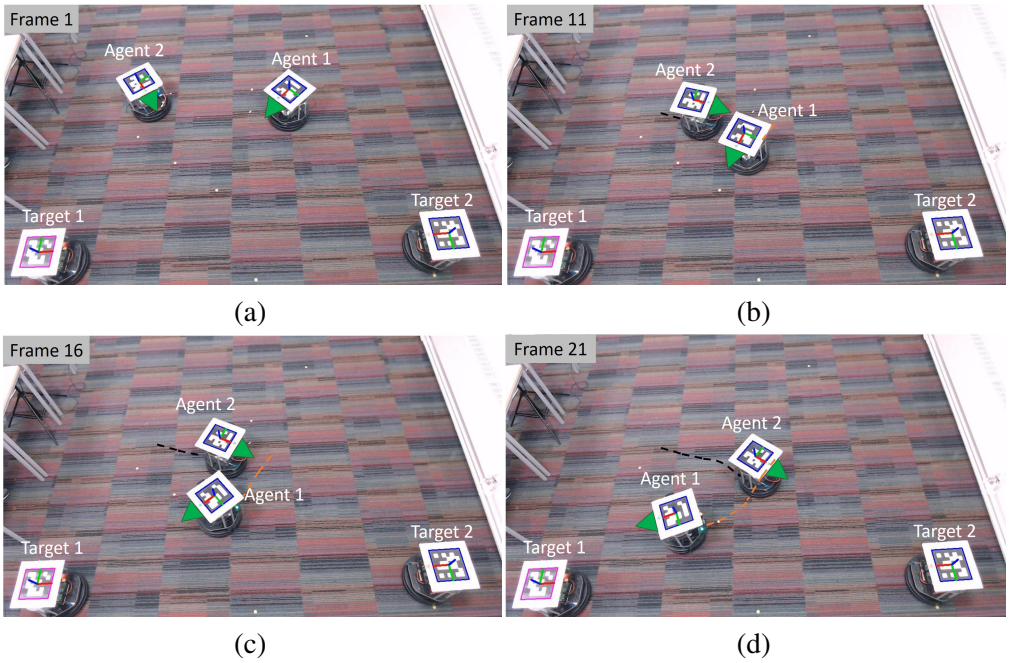


Figure 6.9: Sample frames from static camera 1 in Scenario II. Agent 1 (2) follows target 1 (2). The green triangle highlights the heading direction of each agent. Markers with blue bounding boxes are detected by the detector and markers with purple boxes are detected by the KLT corner tracker. The paths of agent 1 and agent 2 are shown as dashed lines in orange and black, respectively.

6.3.4 Latency analysis

This section analyses the time breakdown for local processing and message transmission latency. The message type is indicated by capital letters, with the subscript S and R indicating the time stamp on sending and receiving, respectively (see Fig. 6.11).

For a static camera, the *sync image capturer* publishes the latest captured image on receiving a trigger message or at the end of a pre-set timer. The measured time stamps correspond to when a *trigger* message is sent from the synchronisation machine, T_S, when the local machine receives the trigger message on time or when the local timer ends in the case of late message arrival, T_R, and when the latest captured image is sent, C_S. The *aruco detector* node publishes the estimated marker pose on receiving the captured images. The time stamps are measured when the captured message is received, C_R, and the pose measurement message is sent, M_S. The *distributed tracker* node, which is the same at static cameras and at agents, initiates the distributed fusion process on receiving the measurement message and publishes the estimated states of agents and targets. The time stamps are measured when the measurement message is received, M_R, and when the state message is sent, S_S.

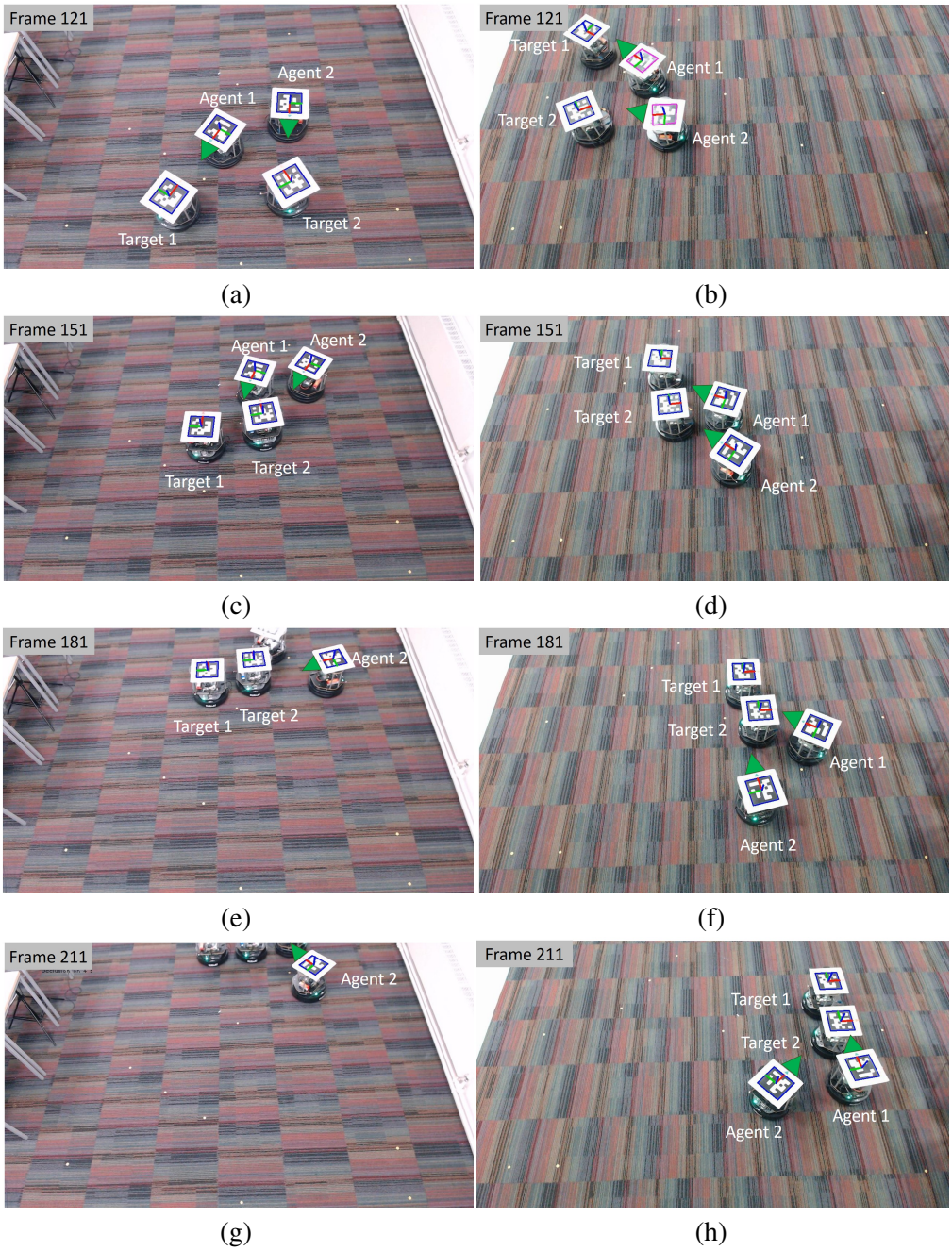


Figure 6.10: Sample frames from static camera 1 (left column) and static camera 3 (right column). The green triangle highlights the heading direction of each agent. Markers with blue bounding boxes are detected by the detector and markers with purple boxes are detected by the KLT corner tracker.

For an agent, the *sync odm capturer* node publishes the latest updated odom measurement from the on-board sensors on receiving the trigger message or at the end of the pre-set timer. The time stamps are measured when the trigger message is sent, T_S, when the trigger message is received or the pre-set timer ends, T_R, and when the latest measurement message is sent, M_S. The *collision-free follower* node computes and publishes the collision-free control message on

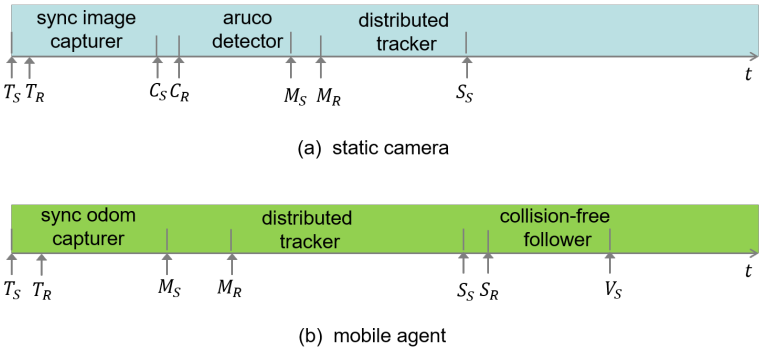


Figure 6.11: Illustration of the timeline for a static camera (blue) and an agent (green) within one cycle. T is the trigger message for synchronised capturing, C is the message for captured sensory data. M is the message for measurements, S for estimated states, and V for robotic control. Subscript S refers to the time stamp when the corresponding message is sent and R when the corresponding message is received.

receiving the states of agents and targets. The time stamps are measured when the state message is received, S_R, and when the velocity control message is sent, V_S.

Fig. 6.12 shows the results of a 100-second test with a system frequency of 1 Hz using three static cameras and one agent to investigate the time cost of each node running on different computing platforms.

$T_S - T_R$ presents the duration from the time when the synchronisation machine starts a new cycle to the time when the local machine also starts the new cycle. The synchronisation machine (XPS laptop) takes negligible time to start a new cycle as the transmission of trigger messages are within the same machine. Instead the time taken by other machines to start a new cycle varies because of the transmission delay between machines, the time is bounded within 0.05 s due to the use of the local timer.

$T_R - C_S$ is the duration between the start of a local new cycle and the time when newly captured sensory data (at static cameras) or measurements (at agents) is sent. A time difference of 0.01 s separates the start of a cycle and the capture of new data or measurements as the capturing process runs as a separate thread at a higher frequency than that of the system. When a new cycle starts, the actual capture may occur slightly later or earlier depending on the multi-thread handling scheme.

$C_R - M_S$ is the duration of the marker detection and pose estimation that takes place at static cameras. The Latitude (with four logical processing units) takes on average around 0.1 s, more than twice the time taken by Alienware and XPS (with eight logical processing units).

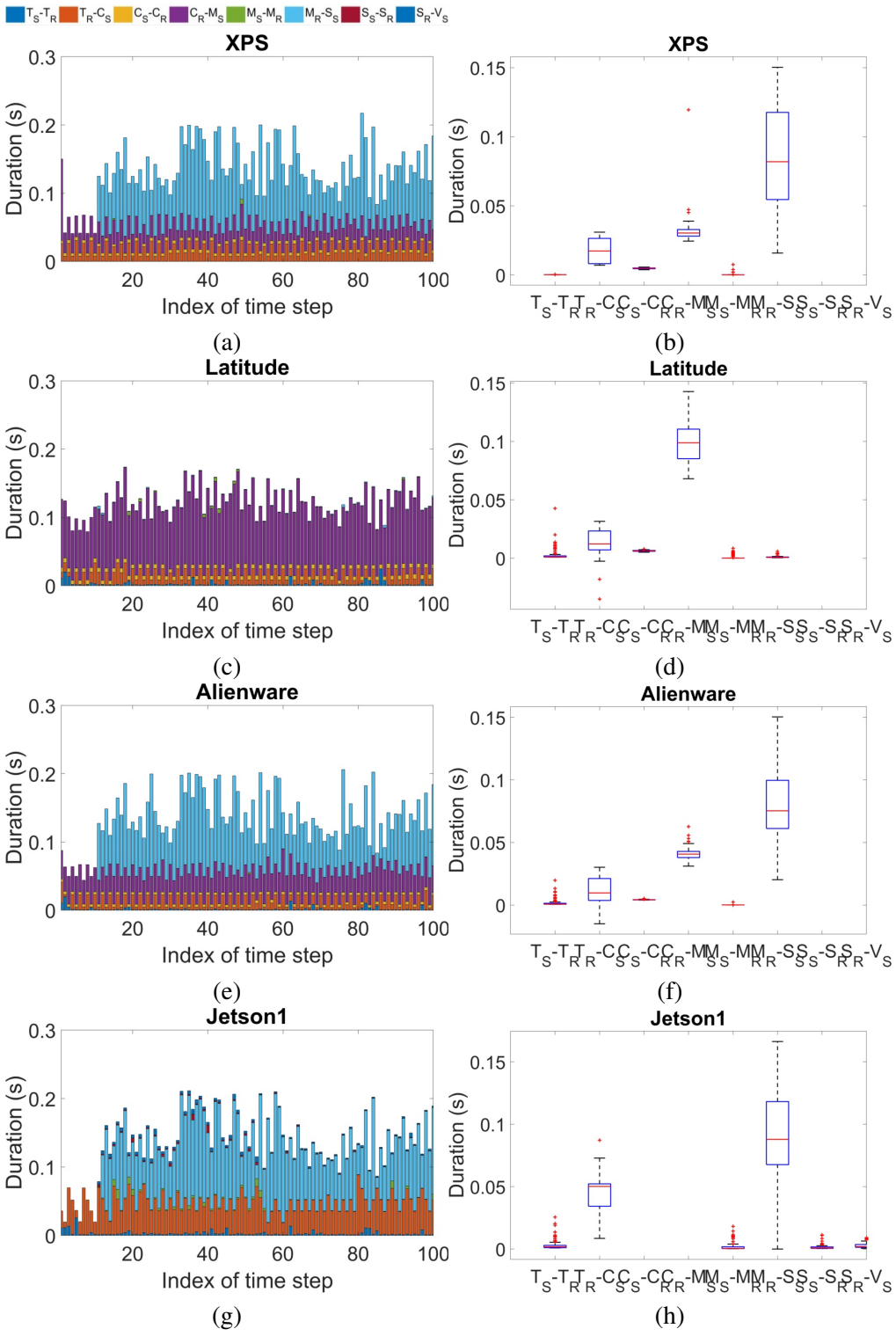


Figure 6.12: Breakdown of the local processing and transmissions over time on three laptops and one Jetson board. Left column: stacked bar plots for the duration breakdown at each machine. Right column: corresponding box plots of all durations. (a),(b) Results for XPS as static camera 1; (c),(d) results for Latitude as static camera 2; (e),(f) results for Alienware as static camera 3; (g),(h) results for the Jetson board as agent 1.

$M_R - S_S$ is the duration for distributed tracking. Each machine waits for the fusion message corresponding to the same round from all neighbouring machines and then performs a round of fusion. The Latitude takes longer in preparing its measurements which leads to late transmission of its *Fusion* message. Each machine performs fusion once the pre-set timer expires in the case of missing or delayed *Fusion* messages. This explains why the $M_R - S_S$ of all machines is bound within the length of the pre-set timer, 0.15 s. Finally, the duration for computing the collision-free control by each agent, $S_R - V_S$, is 0.01 s.

The overall system frequency with the current implementation is 4 Hz. The major contributor to the latency is image processing, which can be improved with more powerful computing devices and parallelisation techniques.

6.4 Conclusion

This Chapter presented the implementation of a distributed multi-agent system where agents perform distributed tracking with ceiling-mounted static cameras and computes collision-free robotic controls without any central processing. The system handles practical issues, including the communication imperfections through the use of local timers and the tracking failures by temporarily stopping the agents. With real robotic platforms, we showed that the system is able to handle collision avoidance among agents and targets while maintaining the view on targets being followed.

As future work, we will speed-up the current frequency, i.e. 4 Hz, of the system with a parallel implementation of the vision pipeline and we will extend the system to work in wider areas with a network that is not fully connected.

Chapter 7

Conclusions

7.1 Summary of achievements

This thesis presented a collaborative tracking framework with static cameras for camera-equipped agents to perform collision-free active visual tracking in wide areas (e.g. in museums or airports). In order to achieve a scalable multi-agent system, agents are designed to track nearby targets and make independent decisions, i.e. selecting their target to serve and computing collision-free control, via only neighbourhood communication without any control centre.

Agents track targets jointly with static ceiling-mounted smart cameras so that multi-view information can be utilised to address occlusions in populated scenes. Distributed tracking enables each node (i.e. a static camera or an agent) with agreed target state estimates via neighbourhood communication. However, traditional distributed tracking schemes are bandwidth demanding as all nodes are involved for the iterative state updates. Communication imperfections, e.g. packets loss and errors, can worsen tracking accuracy as the number of targets increases. This thesis therefore proposed a three-stage communication-aware coalition formation framework prior to distributed tracking. Each static camera joins tracking coalitions based on a marginal utility that accounts for both tracking confidence and the communication link quality. The proposed coalition-based distributed tracking outperforms distributed tracking without coalitions in terms of convergence speed and tracking accuracy. Compared to the decentralised tracking strategy, the coalition-based distributed tracking achieves comparable tracking accuracy and communication cost with the modelling of packet loss and packet errors.

On hearing requests from nearby static cameras, agents select their target based on a proposed local criterion that accounts for the relative agent-target states and the tracking priority provided by static cameras. With the proposed criterion, a higher prioritised observation time can be achieved compared to the distance-based agent-target assignment without consuming more energy. Each agent computes the robotic control to actively track its target with view maintenance. This thesis investigated the trade-off between view maintenance and energy efficiency. With the proposed energy-efficient motion controller, 10% energy cost can be reduced without compensating much deterioration in the view maintenance performance (3%) when tested with real people trajectories, compared to the controller that only accounts for view maintenance.

As there is the presence of multiple agents and targets, collision avoidance is considered in the motion control. The ORCA method is appropriate for collision avoidance among multiple agents with different tasks due to its reciprocity for avoiding undesired oscillations. This thesis proposed an ORCA-based method with adaptive responsibility sharing and heading-aware robotic control to address view maintenance during collision avoidance manoeuvres. The adaptive responsibility sharing algorithm assigns less responsibility to agents with higher risks of losing their target in the camera's FoV based on the difference between their preferred velocity and current velocity. The heading-aware robotic control algorithm aims to avoid unnecessary target loss when an agent has backward movements by minimising the deviation angle from the agent heading to its target in a smooth manner. With real people trajectories extracted from public datasets, the adaptive pair-wise responsibility algorithm demonstrates a slightly better distance maintenance performance, while the heading-aware control mapping algorithm shows a prominent advantage in the deviation angle maintenance, compared to the ORCA-based method for differential-drive agents. The proposed method with two elements achieves $\geq 20\%$ improvement in the deviation angle maintenance and 3% improvement in distance maintenance.

In short, this thesis addressed i) the scalable collaboration among static cameras and agents to enable agents perform autonomous active visual tracking and ii) the collision avoidance among agents and targets with emphasis on view maintenance. In order to deploy such multi-agent system to serve people in populated environments, further research efforts are required where three future directions are identified with respect to the current advances in robotic control and computer vision.

7.2 Future work

Deep learning for multi-agent collision avoidance

The ORCA method uses well-defined geometric rules for collision avoidance which may fail in densely crowded scenes. The strategy that a person employs to navigate in crowded scenes without colliding with other people can be promising to address this issue and in the mean time achieve social awareness [69].

Deep learning techniques, such as recurrent neural networks, can be used for learning implicit social conventions that people obey for collision avoidance using large datasets. Social long short-term memory network (Social LSTM) [3] has been used for human movement prediction. The network was trained using publicly available datasets and the prediction results have been successfully tested on a single robot, Jackrabbot[1], for navigating in a crowded university campus. A neural network has also been trained for avoiding collisions among multiple agents, where the problem is formulated as a multi-class classification problem [79]. The training inputs are the ground-plane observation of nearby agents and their preferred velocities, and the output class is the partition of the velocity space. The network is trained using simulated data generated by the ORCA method. Although the learnt strategy is not as good as ORCA in terms of motion smoothness, this work demonstrates the capability of deep learning techniques for modelling a highly dynamic system. Moreover, it can be time-consuming to anticipate multi-agent paths for optimised collision avoidance. Deep learning techniques, for instance with deep reinforcement learning [26], can reduce the expensive online computation to offline training procedures, so that real-time multi-agent control becomes feasible.

Deep learning is promising, but because it is a data-driven approach, it is difficult to fully interpret the complex non-linear networks. For safety-sensitive applications, the explainability of a model is essential to design more efficient and safer models. Recent research efforts have been put in interpreting deep neural networks [87] by visualising the input that produces the maximal activation either at a neuron level [143] or at a layer level [82]. As future works, in addition to use the deep neural networks, we will also analyse the reasons of its efficiency and the causes of its failures in order to achieve risk-free applications.

Accurate people detection

[1] http://cvgl.stanford.edu/projects/jackrabbot/

For real-world applications, it is essential to achieve reliable and accurate detection of people from the image plane without artificial features (e.g. fiducial markers). Traditionally, hand-crafted features, e.g. gradients (e.g. edges) [30] or their spatial arrangement (e.g. textures) [136], are extracted and trained for an object model using various machine learning algorithms, such as support vector machine [42]. Recent years, deep learning techniques have been largely applied and outperform traditional approaches without the need of explicit feature extraction [45]. A promising object detector, the YOLO (short for You Only Look Once) detector, uses an end-to-end neural network to predict both bounding boxes (localisation) and class probabilities (classification) [109]. YOLO detector can achieve 45 frames per second with low false positive rates but more localisation errors [109]. Different detectors can be further combined to improve the detection accuracy in order to complement the weakness of any single detector [126, 115]. The main challenge we see is the efficient combination of multiple detectors that improves the detection accuracy while achieving real-time on-board processing.

Simultaneous localisation and mapping for flexible deployments

This thesis assumes agents are aware of their own poses and the environment is controlled without any static obstacles. For the deployment of real robots in uncontrolled environments, robots are required to be able to accurately localise themselves and form local understanding of the environment, i.e. the obstacles around them. SLAM works on the simultaneous estimation of the robot state and the construction of the environment in a global coordinate, using on-board sensors, such as a camera or range finder [19]. SLAM using visual inputs (vision-based SLAM) can provide sufficient localisation accuracy, even for commercial usage, such as Dyson's vacuum robot[2]. Recently, multiple-robot SLAM has attracted research attention for the benefits of its improved speed of building map in wide areas and robustness to single-robot failures. However price comes with the increased complexity in map updating [114]. Centralised paradigms have been mostly adopted for collaborative vision-based SLAM due to limited on-board computational capability [140, 120], where each robot independently performs real-time visual odometry, sends map points and key frames to a more computationally powerful central server for map updating, and receives the merged map from the server. As future work, we would like to incorporate collaborative

[2]http://spectrum.ieee.org/automaton/robotics/home-robots/dyson-the-360-eye-robot-vacuum

vision-based SLAM to enable robots navigate in less controlled environments and explore a distributed paradigm that avoids the use of a central server.

Bibliography

[1] K. Abas, C. Porto, and K. Obraczka. Wireless smart camera networks for the surveillance of public spaces. *Computer*, 47(5):37–44, May 2014.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, Aug 2002.

[3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F. Li, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 961–971, Las Vegas, US, Jun 2016.

[4] J. Alonso-Mora, S. Baker, and D. Rus. Multi-robot navigation in formation via sequential convex programming. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 4634–4641, Hamburg, Germany, Sep 2015.

[5] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart. Reciprocal collision avoidance for multiple car-like robots. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 360–366, Saint Paul, US, May 2012.

[6] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Proc. of Int'l Symposium on Distributed Autonomous Robotic Systems*, Lausanne, Switzerland, Nov 2010.

[7] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley. Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots*, 39:101–121, Jan 2015.

[8] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 50(2):174–188, Feb 2002.

[9] I. Ashokaraj, A. Tsourdos, P. Silson, and B. A. White. Sensor based robot localisation and navigation: using interval analysis and unscented kalman filter. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 7–12, Sendai, Japan, Sep 2004.

[10] J. Banfi, J. Guzzi, A. Giusti, L. Gambardella, and G. A. D. Caro. Fair multi-target track-

ing in cooperative multi-robot systems. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 5411–5418, Seattle, US, May 2015.

[11] D. Bareiss and J. V. D. Berg. Generalized reciprocal collision avoidance. *Int'l J. of Robotics Research*, 34(12):1501–1514, Oct 2015.

[12] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Proc. of Int'l Symp. on Robotics Research*, volume 70, pages 3–19. Springer, 2009.

[13] A. S. Bernabe, J. R. M. de Dios, and A. Ollero. Efficient cluster-based tracking mechanisms for camera-based wireless sensor networks. *IEEE Trans. on Mobile Computing*, Nov 2015.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. on Network*, 14(SI):2508–2530, Jun 2006.

[15] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *Computer*, 39(2):68–75, Feb 2006.

[16] M. Bramberger, M. Quaritsch, T. Winkler, B. Rinner, and H. Schwabach. Integrating multi-camera tracking into a dynamic task allocation system for smart cameras. In *Proc. of IEEE Conf Advanced Video and Signal Based Surveillance*, pages 474–479, Como, Italy, Sep 2005.

[17] A. E. Bryson. Optimal control1950 to 1985. *IEEE Control Systems*, 16(3):26–33, Jun 1996.

[18] A. L. Bustamante, J. M. Molina, and M. A. Patricio. A practical approach for active camera coordination based on a fusion-driven multi-agent system. *Int'l J. of Systems Science*, 45(4):741–755, Apr 2014.

[19] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. on Robotics*, 32(6):1309–1332, Dec 2016.

[20] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber. Collision avoidance for multiple agent systems. In *Proc. of IEEE Conf. on Decisions and Control*, pages 539–543, Hawaii, US, Dec 2003.

[21] T. H. Chang and S. Gong. Tracking multiple people with a multi-camera system. In *Proc. of IEEE Workshop on Multi-Object Tracking*, pages 19–26, Vancouver, Canada, Jul 2001.

[22] K. Charalampous, I. Kostavelis, and A. Gasteratos. Recent trends in social aware robot navigation: A survey. *Robotics and Autonomous Systems*, 93:85 – 104, 2017.

[23] C. Chen, Yi Yao, D. Page, B. Abidi, A. Koschan, and M. Abidi. Camera handoff with adaptive resource management for multi-camera multi-target surveillance. In *Proc. of IEEE Int'l Conf. on Advanced Video and Signal Based Surveillance*, pages 79–86, Santa Fe, US, Sep 2008.

[24] M. Chen, S. Gonzalez, and V. C. M. Leung. Applications and design issues for mobile agents in wireless sensor networks. *IEEE Wireless Communications*, 14(6):20–26, Dec 2007.

[25] S. Y. Chen. Kalman filter for robot vision: A survey. *IEEE Trans. on Industrial Electronics*, 59(11):4409–4420, Nov 2012.

[26] Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized non-communicating multi-agent collision avoidance with deep reinforcement learning. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 285–292, May 2017.

[27] F. Chenavier and J. L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 2588–2593, Nice, France, May 1992.

[28] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. *CoRR*, Jan 2017.

[29] S. Curtis, B. Zafar, A. Gutub, and D. Manocha. Right of way: Asymmetric agent interactions in crowds. *Visual Computer*, 29(12):1277–1292, 2013.

[30] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 886–893, San Diego, US, Jun 2005.

[31] A. J. Davison and N. Kita. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 384–391, Kauai, US, Dec 2001.

[32] T. Dewi, N. Uchiyama, and S. Sano. Service mobile robot control for tracking a moving object with collision avoidance. In *Proc. of IEEE Int'l Workshop on Advanced Robotics and its Social Impacts*, pages 1–6, Lyon, France, Jun 2015.

[33] B. Dieber, L. Esterle, and B. Rinner. Distributed resource-aware task assignment for complex monitoring scenarios in visual sensor networks. In *Proc. of ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, pages 1–6, Oct 2012.

[34] C. Ding, B. Song, A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury. Collaborative sensing in a distributed ptz camera network. *IEEE Trans. on Image Processing*, 21(7):3282–3295, Jul 2012.

[35] Y. Ding, M. Zhu, Y. He, and J. Jiang. P-cmommt algorithm for the cooperative multi-robot observation of multiple moving targets. In *Proc. of World Congress on Intelligent Control and Automation*, pages 9267–9271, Dalian, China, Jun 2006.

[36] J. J. DiSteffano, A. R. Stubberud, and I. J. Williams. *Feedback and control systems*. McGraw-Hill, 1967.

[37] G. Doisy, A. Jevtic, E. Lucet, and Y. Edan. Adaptive person-following algorithm based on depth images and mapping. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems Workshop on Robot Motion Planning*, Vilamoura-Algarve, Portugal, Oct 2012.

[38] P. Dollar, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, Aug 2014.

[39] X. Dong, X. Dong, and J. Dong. *Monocular Visual-IMU Odometry: A Comparative Evaluation of the Detector-Descriptor Based Methods*, pages 81–95. Springer, Amsterdam, Netherlands, Oct 2016.

[40] A. O. Ercan, A. E. Gamal, and L. J. Guibas. Object tracking in the presence of occlusions using multiple cameras: A sensor network approach. *ACM Trans. of Sensor Netwwork*, 9(2):16:1–16:36, Apr 2013.

[41] L. Esterle, P. R. Lewis, M. Bogdanski, B. Rinner, and Y. Xin. A socio-economic approach to online vision graph generation and handover in distributed smart camera networks. In *Proc. of ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, pages 1–6, Gent, Belgium, Aug 2011.

[42] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sep 2010.

[43] P. Fiorini and Z. Shillert. Motion planning in dynamic environments using velocity obstacles. *Int'l Journal of Robotics Research*, 17:760–772, 1998.

[44] S. Garrido-Jurado, R. Muoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marn-Jimnez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6), 2014.

[45] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 580–587, Columbus, US, Jun 2014.

[46] R. Gockley, J. Forlizzi, and R. Simmons. Natural person-following behavior for social robots. In *Procs. of ACM/IEEE Int'l Conf. on Human-robot Interaction*, pages 17–24, New York, US, 2007.

[47] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan. Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons. In *Proc. ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, pages 219–226, Vienna, Austria, Sep 2007.

[48] C. Granata and P. Bidaud. A framework for the design of person following behaviors for social mobile robots. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 4652–4659, Vilamoura-Algarve, Portugal, Oct 2012.

[49] A. E. Guevara, A. Hoak, J. T. Bernal, and H. Medeiros. Vision-based self-contained target following robot using bayesian data fusion. In *Proc. of Int'l Symp. on Visual Computing*, pages 1–12, Las Vegas, US, Dec 2016.

[50] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. on Information Theory*, 46(2):388–404, Mar 2000.

[51] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls. Multi-robot collision avoidance with localization uncertainty. In *Proc. of Int'l Conf. on Autonomous Agents and Multiagent Systems*, pages 147–154, Valencia, Spain, 2012.

[52] K. Hirose, D. Chugo, S. Yokota, and K. Takase. Service robots navigation using pictographs detection for indoor environment. In *Proc. of IEEE Annual Conf. on Industrial Electronics Society*, pages 2170–2175, Melbourne, Australia, Nov 2011.

[53] O. Hlinka, O. Sluciak, F. Hlawatsch, and M. Rupp. Distributed data fusion using iterative

covariance intersection. In *Proc. of IEEE Int'l Conf. on Acoustics, Speech and Signal Processing*, pages 1861–1865, Florence, Italy, May 2014.

[54] N. Ilic, M. S. Stankovic, and S. S. Stankovic. Adaptive consensus-based distributed target tracking in sensor networks with limited sensing range. *IEEE Trans. on Control Systems Technology*, 22(2):778–785, Mar 2014.

[55] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Computing Research Repository*, 1998.

[56] B. Jensen, N. Tomatis, L. Mayor, A. Drygajlo, and R. Siegwart. Robots meet humans-interaction in public spaces. *IEEE Trans. on Industrial Electronics*, 52(6):1530–1546, Dec 2005.

[57] W. Jeong and K. Lee. Cv-slam: a new ceiling vision-based slam technique. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 3195–3200, Edmonton, Canada, Aug 2005.

[58] A.T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information weighted consensus. In *Proc. of IEEE Conf. on Decision and Control*, pages 2732–2737, Melbourne, Australia, Dec 2012.

[59] A.T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information consensus for distributed multi-target tracking. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2403–2410, Portland,Oregon, Jun 2013.

[60] S. Kar and J. M. F. Moura. Sensor networks with random links: Topology design for distributed consensus. *IEEE Trans. on Signal Processing*, 56(7):3315–3326, Jul 2008.

[61] S. Katragadda and A. Cavallaro. Neighbour consensus for distributed visual tracking. In *Proc. of IEEE Int'l Conf. on Intelligent Sensors, Sensor Networks and Information Processing*, pages 1–6, Singapore, Apr 2015.

[62] S. Katragadda, J. C. SanMiguel, and A. Cavallaro. Consensus protocols for distributed tracking in wireless camera networks. In *Proc. of Int'l Conf. on Information Fusion*, pages 1–8, Salamanca, Spain, Jul 2014.

[63] A. Khan, B. Rinner, and A. Cavallaro. Multiscale observation of multiple moving targets using micro aerial vehicles. In *Proc. in IEEE Int'l Conf. on Intelligent Robots and Systems*, pages 4642–4649, Hamburg, Germany, Sep 2015.

[64] M. E. Khan. Matrix inversion lemma and information filter. Technical report, Bangalore, India, 2005.

[65] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int'l J. of Robotics Research*, 5(1):90–98, Mar 1986.

[66] C. J. Kim and D. Chwa. Obstacle avoidance method for wheeled mobile robots using interval type-2 fuzzy neural network. *IEEE Trans. on Fuzzy Systems*, 23(3):677–687, Jun 2015.

[67] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia. People tracking and following with mobile robot using an omnidirectional camera and a laser. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 557–562, Orlando, US, May 2006.

[68] A. Kolling and S. Carpin. Multirobot cooperation for surveillance of multiple moving targets - a new behavioral approach. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 1311–1316, Orlando, US, May 2006.

[69] T. Kruse, A. P. Kumar, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, Dec 2013.

[70] T. P. Lambrou and P. G. Christos. Collaborative path planning for event search and exploration in mixed sensor networks. *Int'l Jornal on Robotics Research*, 32(12):1424–1437, Oct 2013.

[71] J. P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In *Robot motion planning and control*, pages 1–53. Springer, 1998.

[72] S. M. LaValle, H. H. Gonzalez-Banos, C. Becker, and J. C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 731–736, Albuquerque, US, Apr 1997.

[73] S. Lee, Y. Cho, M. H. Bo, B. You, and S. Oh. A stable target-tracking control for unicycle mobile robots. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, volume 3, pages 1822–1827, Takamatsu, Japan, Nov 2000.

[74] Y. Li and B. Bhanu. A comparison of techniques for camera selection and handoff in a video network. In *Proc. of ACM/IEEE Int'l Conf. on Distributed Smart Cameras*, pages 1–8, Como, Italy, Aug 2009.

[75] Y. Li and B. Bhanu. Utility-based camera assignment in a video network: A game theoretic framework. *IEEE Sensors J.*, 11(3):676–687, Mar 2011.

[76] L. Liang, Z. Xi, and M. Huadong. Optimal node selection for target localization in wireless camera sensor networks. *IEEE Trans. on Vehicular Technology*, 59(7):3562–3576, Sep 2010.

[77] F. Lin, X. Dong, B. M. Chen, K. Y. Lum, and T. H. Lee. A robust real-time embedded vision system on an unmanned rotorcraft for ground target following. *IEEE Trans. on Industrial Electronics*, 59(2):1038–1049, Feb 2012.

[78] S. Liu and D. Sun. Minimizing energy consumption of wheeled mobile robots via optimal motion planning. *IEEE/ASME Trans. on Mechatronics*, 19(2):401–411, Apr 2014.

[79] P. Long, W. Liu, and J. Pan. Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics and Automation Letters*, 2(2):656–663, Apr 2017.

[80] W. Luo, X. Zhao, and T. Kim. Multiple object tracking: A review. *CoRR*, 2014.

[81] R. Madhavan, K. Fregene, and L. E. Parker. Distributed heterogeneous outdoor multi-robot localization. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, volume 1, pages 374–381, Washington, US, May 2002.

[82] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, pages 1–23, 2016.

[83] A. Martinelli, N. Tomatis, and R. Siegwart. Simultaneous localization and odometry self calibration for mobile robot. *Autonomous Robots*, 22(1):75–85, Jan 2007.

[84] H. Medeiros, J. Park, and A.C. Kak. Distributed object tracking using a cluster-based kalman filter in wireless camera networks. *IEEE J. of Selected Topics in Signal Processing*, 2(4):448–463, Aug 2008.

[85] Y. Mei, Y. Lu, Y. C. Hu, and C. S. G. Lee. Energy-efficient motion planning for mobile robots. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 4344–4349, New Orleans, US, Apr 2004.

[86] Y. Mei, Y. Lu, C. S. G. Lee, and Y. C. Hu. Energy-efficient mobile robot exploration. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 505–511, Orlando, US, May 2006.

[87] G. Montavon, W. Samek, and K.R. Müller. Methods for interpreting and understanding deep neural networks. *CoRR*, abs/1706.07979, 2017.

[88] K. Morioka, J. Lee, and H. Hashimoto. Human-following mobile robot in a distributed intelligent sensor network. *IEEE Trans. on Industrial Electronics*, 51(1):229–237, Feb 2004.

[89] R. Murrieta-Cid, H. H. Gonzalez-Banos, and B. Tovar. A reactive motion planner to maintain visibility of unpredictable targets. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 4242–4248, Washington, US, May 2002.

[90] T. P. Nascimento, A. P. Moreira, and G. S. C. Andre. Multi-robot nonlinear model predictive formation control: Moving target and target absence. *Robotics and Autonomous Systems*, 61(12):1502 – 1515, 2013.

[91] T. Naseer, J. Sturm, and D. Cremers. Followme: Person following and gesture recognition with a quadrocopter. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 624–630, Tokyo, Japan, Nov 2013.

[92] P. Natarajan, A. K. Pradeep, and M. Kankanhalli. Multi-camera coordination and control in surveillance systems: A survey. *ACM Trans. on Multimedia Computation Communication Application*, 11(4):57:1–57:30, June 2015.

[93] W. Niehsen. Information fusion based on fast covariance intersection filtering. In *Proc. of Int'l Conf. on Information Fusion.*, volume 2, pages 901–904, Annapolis, US, Jul 2002.

[94] P. Oettershagen, A. Melzer, S. Leutenegger, K. Alexis, and R. Siegwart. Explicit model predictive control and l1-navigation strategies for fixed-wing uav path tracking. In *Proc. of Mediterranean Conf. on Control and Automation*, pages 1159–1165, June 2014.

[95] R. Olfati-Saber. Distributed Kalman filter with embedded consensus filters. In *Proc. of IEEE Conf. on Decision and Control*, pages 8179–8184, Seville, Spain, Dec 2005.

[96] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proc. of the IEEE*, 95(1):215–233, Jan 2007.

[97] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Automatic Control*, 49(9):1520–1533, Sep 2004.

[98] D. Panagou, M. Turpin, and V. Kumar. Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple Lyapunov functions approach. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 6757–6762, Hong Kong, May 2014.

[99] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255, May 2002.

[100] L. E. Parker and B. A. Emmons. Cooperative multi-robot observation of multiple moving targets. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 2082–2089, Albuquerque, US, Apr 1997.

[101] M. A. Patricio, J. Carbó, O. Pérez, J. García, and J. M. Molina. Multi-agent framework in visual sensor networks. *EURASIP J. on Advances in Signal Processing*, 2007(1):1–21, Nov 2006.

[102] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy. Computer vision based general object following for gps-denied multirotor unmanned vehicles. In *Proc. of American Control Conf.*, pages 1886–1891, Portland, US, Jun 2014.

[103] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella. Performance study of ieee 802.15.4 using measurements and simulations. In *Proc. of IEEE Wireless Communications and Networking Conf.*, pages 487–492, Las Vegas, US, Apr 2006.

[104] D. Pizarro, M. Mazo, E. Santiso, M. Marron, D. Jimenez, S. Cobreces, and Cristina Losada. Localization of mobile robots using odometry and an external vision sensor. *Sensors*, 10(4):3655–3680, 2010.

[105] S. A. P. Quintero, D. A. Copp, and J. P. Hespanha. Robust uav coordination for target tracking using output-feedback model predictive control with moving horizon estimation. In *Proc. of American Control Conference*, pages 3758–3764, July 2015.

[106] F. Qureshi and D. Terzopoulos. Distributed coalition formation in visual sensor networks: A virtual vision approach. In *Distributed Computing in Sensor Systems*, volume 4549 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2007.

[107] R. Rahman, M. Alanyali, and V. Saligrama. Distributed tracking in multihop sensor networks with communication delays. *IEEE Trans. on Signal Processing*, 55(9):4656–4668, Sep 2007.

[108] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems*, 20(3):38–52, Jun 2000.

[109] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, 2015.

[110] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Trans. on Robotics and Automation*, 8(5):501–518, Oct 1992.

[111] B. Rinner and W. Wolf. An introduction to distributed smart cameras. *Proc. of the IEEE*, (10):1565–1575, Oct 2008.

[112] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. *CoRR*, 09, 2016.

[113] H. Sabbineni and K. Chakrabarty. Location-aided flooding: an energy-efficient data dissemination protocol for wireless-sensor networks. *IEEE Trans. on Computers*, 54(1):36–46, Jan 2005.

[114] S. Saeedi, M. Trentini, M. Seto, and H. Li. Multiple-robot simultaneous localization and mapping: A review. *J. Field Robot*, 33(1):3–46, Jan 2016.

[115] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro. *Online Multi-target Tracking with Strong and Weak Detections*, pages 84–99. Springer, Amsterdam, Oct 2016.

[116] J. C. SanMiguel and A. Cavallaro. Cost-aware coalitions for collaborative tracking in resource-constrained camera networks. *IEEE Sensors J.*, 15(5):2657–2668, May 2015.

[117] J. C. SanMiguel, C. Micheloni, K. Shoop, G. L. Foresti, and A. Cavallaro. Self-reconfigurable smart camera networks. *IEEE Computer*, 47(5):67–73, May 2014.

[118] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011.

[119] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Worz. Vision based person tracking with a mobile robot. In *Proc. of British Machine Vision Conf.*, pages 418 – 427, Southampton, UK, Sep 1998.

[120] P. Schmuck and M. Chli. Multi-uav collaborative monocular slam. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 3863–3870, Singapore, May 2017.

[121] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Int'l J. of Robotics Research*, 21(8):735–758, 2002.

[122] T. B. Sheridan. Human-robot interaction. *Human Factors*, 58(4):525–532, 2016.

[123] Z. Shiller, F. Large, and S. Sekhavat. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 3716–3721, Seoul, Korea, Jul 2001.

[124] D. Simon. *Nonlinear Kalman filtering*, pages 393–431. John Wiley, 2006.

[125] D. Simon. *The unscented Kalman filter*, pages 433–459. John Wiley, 2006.

[126] F. D. Smedt and T. Goedem. Open framework for combined pedestrian detection. In *Proc. of Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 551–558, Berlin, Germany, 2015.

[127] J. Snape, J. V. D. Berg, S. J. Guy, and D. Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 5917–5922, St. Louis, US, Oct 2009.

[128] J. Snape, J. V. D. Berg, S. J. Guy, and D. Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 4584–4589, Taipei, Taiwan, Oct 2010.

[129] L. Tessens, M. Morbee, H. Aghajan, and W. Philips. Camera selection for tracking in distributed smart camera networks. *ACM Trans. on Sensor Networks*, 10(2):1–33, Jan 2014.

[130] C. Teuliere, L. Eck, and E. Marand. Chasing a moving target from a flying uav. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 4929–4934, San Francisco, US, Sep 2011.

[131] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hhnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *Int'l J. of Robotics Research*, 19(11):972–999, 2000.

[132] P. Tokekar, V. Isler, and A. Franchi. Multi-target visual tracking with aerial robots. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 3067–3072, Chicago, US, Sep 2014.

[133] D. Ustebay, M. Coates, and M. Rabbat. Distributed auxiliary particle filters using selective

gossip. In *Proc. of IEEE Int'l Conf. on Acoustics, Speech and Signal Processing*, pages 3296–3299, Prague, Czech Republic, May 2011.

[134] H. Y. Wang, S. Itani, T. Fukao, and N. Adachi. Image-based visual adaptive tracking control of nonholonomic mobile robots. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, volume 1, pages 1–6, Maui, US, Oct 2001.

[135] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, 34(1):3 – 19, 2013.

[136] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *Proc. of IEEE Int'l Conf. on Computer Vision*, pages 32–39, Kyoto, Japan, Sep 2009.

[137] H. Wei, W. Lu, P. Zhu, G. Huang, J. Leonard, and S. Ferrari. Optimized visibility motion planning for target tracking and localization. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 76–82, Chicago, US, Sep 2014.

[138] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, US, 1995.

[139] D. Wilkie, J. V. D. Berg, and D. Manocha. Generalized velocity obstacles. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 5573–5578, St. Louis, US, Oct 2009.

[140] R. Williams, B. Konev, and F. Coenen. Scalable distributed collaborative tracking and mapping with micro aerial vehicles. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 3092–3097, Hamburg, Germany, Sep 2015.

[141] C. Ye, Y. Zheng, S. Velipasalar, and M. C. Gursoy. Energy-aware and robust task (re)assignment in embedded smart camera networks. In *Proc. of IEEE Int'l Conf. on Advanced Video and Signal Based Surveillance*, pages 123–128, Krakow, Poland, Aug 2013.

[142] T. Yoshimi, M. Nishiyama, T. Sonoura, H. Nakamoto, S. Tokura, H. Sato, F. Ozaki, N. Matsuhira, and H. Mizoguchi. Development of a person following robot with vision based target detection. In *Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 5286–5291, Beijing, China, Oct 2006.

[143] J. Yosinski, J. Clune, A. M. Nguyen, T. J. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015.

[144] G. Younes, D. C. Asmar, and E. A. Shammas. A survey on non-filter-based monocular visual SLAM systems. *CoRR*, 2016.

[145] S. Zhu, D. Wang, and C. Low. Ground target tracking using uav with input constraints. *J. of Intelligent and Robotic Systems*, 69(1-4):417–429, 2013.